T.C.
REPUBLIC OF TURKEY
HACETTEPE UNIVERSITY
GRADUATE SCHOOL OF HEALTH SCIENCES

NOVEL STATISTICAL APPROACHES FOR SURVIVAL ANALYSIS OF RNA-SEQUENCING DATA

Ahu CEPHE

Program of Biostatistics

DOCTOR OF PHILOSOPHY THESIS

ANKARA

2024

T.C.
REPUBLIC OF TURKEY
HACETTEPE UNIVERSITY
GRADUATE SCHOOL OF HEALTH SCIENCES

NOVEL STATISTICAL APPROACHES FOR SURVIVAL ANALYSIS OF RNA-SEQUENCING DATA

Ahu CEPHE

Program of Biostatistics

DOCTOR OF PHILOSOPHY THESIS

ADVISOR OF THE THESIS

Prof. Dr. Erdem KARABULUT

CO-ADVISOR

Assoc. Prof. Dr. Gökmen ZARARSIZ

ANKARA

2024

NOVEL STATISTICAL APPROACHES FOR SURVIVAL ANALYSIS Of RNA-SEQUENCING DATA

Ahu CEPHE

Supervisor: Prof. Dr. Erdem KARABULUT

Co-supervisor: Assoc. Prof. Dr. Gökmen ZARARSIZ

This thesis study has been approved and accepted as a PhD dissertation in "Biostatistics Program" by the assesment committee, whose members are listed below, on March 26,2024.

Chairman of the Committee :	***Prof. Dr. Pınar ÖZDEMİR***
	Hacettepe University
Member :	***Assoc. Prof. Dr. S. Kenan KÖSE***
	Ankara University
Member :	***Assoc. Prof. Dr. Osman DAĞ***
	Hacettepe University
Member :	***Assist. Prof. Dr. Sevilay KARAHAN***
	Hacettepe University
Member :	***Assist. Prof. Dr. Dinçer GÖKSÜLÜK***
	Erciyes University

This dissertation has been approved by the above committee in conformity to the related issues of Hacettepe University Graduate Education and Examination Regulation.

16 Nisan 2024

Prof. Müge YEMİŞCİ ÖZKAN, MD, PhD

Director

YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezimin/raporumun tamamını veya herhangi bir kısmını, basılı (kağıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma iznini Hacettepe Üniversitesine verdiğimi bildiririm. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları bana ait olacaktır.

Tezin kendi orijinal çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alınarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim Kurulu tarafından yayınlanan "***Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge***" kapsamında tezim aşağıda belirtilen koşullar haricince YÖK Ulusal Tez Merkezi / H.Ü. Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

- Enstitü / Fakülte yönetim kurulu kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren 2 yıl ertelenmiştir. [(1)]
- Enstitü / Fakülte yönetim kurulunun gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren ... ay ertelenmiştir. [(2)]
- Tezimle ilgili gizlilik kararı verilmişti

17/04/2024

Ahu CEPHE

---

[1] "Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"

*(1) Madde 6. 1. Lisansüstü tezle ilgili patent başvurusu yapılması veya patent alma sürecinin devam etmesi durumunda, tez **danışmanının** önerisi ve **enstitü anabilim dalının** uygun görüşü üzerine **enstitü** veya **fakülte yönetim kurulu** iki yıl süre ile tezin erişime açılmasının ertelenmesine karar verebilir.*

*(2) Madde 6. 2. Yeni teknik, materyal ve metotların kullanıldığı, henüz makaleye dönüşmemiş veya patent gibi yöntemlerle korunmamış ve internetten paylaşılması durumunda 3. şahıslara veya kurumlara haksız kazanç imkanı oluşturabilecek bilgi ve bulguları içeren tezler hakkında tez **danışmanının** önerisi ve **enstitü anabilim dalının** uygun görüşü üzerine **enstitü** veya **fakülte yönetim kurulunun** gerekçeli kararı ile altı ayı aşmamak üzere tezin erişime açılması engellenebilir.*

(3) Madde 7. 1. Ulusal çıkarları veya güvenliği ilgilendiren, emniyet, istihbarat, savunma ve güvenlik, sağlık vb. konulara ilişkin lisansüstü tezlerle ilgili gizlilik kararı, **tezin yapıldığı kurum** tarafından verilir *. Kurum ve kuruluşlarla yapılan işbirliği protokolü çerçevesinde hazırlanan lisansüstü tezlere ilişkin gizlilik kararı ise, **ilgili kurum ve kuruluşun önerisi** ile **enstitü** veya **fakültenin** uygun görüşü üzerine **üniversite yönetim kurulu** tarafından verilir. Gizlilik kararı verilen tezler Yükseköğretim Kuruluna bildirilir. Madde 7.2. Gizlilik kararı verilen tezler gizlilik süresince enstitü veya fakülte tarafından gizlilik kuralları çerçevesinde muhafaza edilir, gizlilik kararının kaldırılması halinde Tez Otomasyon Sistemine yüklenir

* Tez **danışmanının** önerisi ve **enstitü anabilim dalının** uygun görüşü üzerine **enstitü veya fakülte yönetim kurulu tarafından karar verilir**.

ETHICAL DECLARATION

In this thesis study, I declare that all the information and documents have been obtained in the base of the academic rules and all audio-visual and written information and results have been presented according to the rules of scientific ethics. I did not do any distortion in data set. In case of using other works, related studies have been fully cited in accordance with the scientific standards. I also declare that my thesis study is original except cited references. It was produced by myself in consultation with supervisor Prof. Dr. Erdem Karabulut and written according to the rules of thesis writing of Hacettepe University Institute of Health Sciences .

Ahu CEPHE

ACKNOWLEDGMENTS

I am thankful to my supervisor, Prof. Dr. Erdem Karabulut, whose encouragement, guidance, and support from beginning to end enabled me to understand the subject. I would like to a special thank my second advisor Assoc. Prof. Dr. Gökmen Zararsız, has guided me with his experiences and always spared time for me even in his busy work schedule, not only in performing this study but also in my entire academic journey.

I would like to thank the other thesis committee members, Asst. Prof. Dr. Sevilay Karahan and Asst. Prof. Dr. Dinçer Göksülük for providing valuable suggestions and contribution to the finalization of my thesis. Also, I would like to thank Asst. Prof. Dr. Dinçer Göksülük shared his knowledge and experience in his work and provided technical and hardware assistance for me to use the workstations effectively.

I would like to thank Dr. Necla Koçhan for her support and valuable contributions to many study steps. I would also like to thank them for their support in accessing the datasets used in the study and making language corrections. Despite the intense and challenging education process, I would like to thank Ahmet Sezgin for his support and help while creating the R package. I would like to thank Prof. Dr. Anne-Laure Boulesteix and Erin Craig for patiently answering my questions when I had problems with the study.

I would like to thank Prof. Dr. Reha Alpar, Prof. Dr. Ahmet Öztürk, Prof. Dr. Ergün Karaağaoğlu, Prof. Dr. Pınar Özdemir for making important contributions to me during my master's and doctorate education. I also would like to thank Assoc. Prof. Gözde Ertürk Zararsız and Asst. Serra İlayda Yerlitaş for their friendship, encouragement and motivation. I would like to thank Menekşe Tarla for her support in helping me to carry out the processes easily during my doctoral education.

Finally, I would like to thank my family and my twins for their endless patience, without them this journey would not have been possible.

ABSTRACT

Cephe, A., Novel Statistical Approaches For Survival Analysis Of Rna-Sequencing Data, Hacettepe University Graduate School of Health Sciences, Department of Biostatistics Doctor of Philosophy Thesis, Ankara, 2024. The number of people with cancer is increasing daily, and the mortality for cancer is constantly increasing since the biomarkers of many cancer types are unknown. Also, cancer doesn't progress between individuals similarly, and all patients vary in response to the same treatment because of genetic differences. At this stage, it is very important to apply more effective treatments by making more accurate prognosis predictions using personalized medicine strategies. Estimating survival in cancer patients using survival time provides essential results. With the development of omics technologies, the relationship between survival time and gene expression profiles of patients can now be modeled. RNA-sequencing technology has been used in recent years for survival analysis omics-based due to its advantages. Although RNA-sequencing has many advantages, it differs from classical survival data with high-dimensionality, heterogeneity, and highly-correlated genes. Due to these problems, the regularized Cox methods and machine learning algorithms adapted to survival data are used instead of classical survival algorithms. However, the regularized Cox methods require some assumptions to be met using the Cox algorithm. Machine learning algorithms that are first created for classification problems and then adapted to survival data require additional time and effort. This study aims to develop new approaches that can be used in the survival analysis of RNA-sequencing data by combining voom transformation, stacking algorithm, and lasso methods with block structure. For this purpose, survival data can be converted into binary classification data with the stacking algorithm. Using the sample weights obtained after the voom transformation in priority-Lasso and IPF-Lasso algorithms, two new approaches are presented: voomStackPrio and voomStackIPF. Our approaches were applied to 12 real RNA-sequencing data from the TCGA database. Performance comparisons were made with other survival algorithms in the literature using Harrell's concordance index. The results showed that the performance of the two new approaches was similar or better than other survival algorithms.

Key Words: survival, RNA-sequencing, voom, stacking, IPF-Lasso

ÖZET

Cephe, A., RNA-Dizileme Verilerinin Sağkalım Analizlerinde Yeni İstatistiksel Yaklaşımlar, Hacettepe Üniversitesi Sağlık Bilimleri Enstitüsü Biyoistatistik Programı Doktora Tezi, Ankara, 2024. Kansere yakalanan insanların sayısı her geçen gün artmaktadır ve birçok kanser türüne ait biyobelirteçler bilinemediği için bu hastalıktan ölüm oranları da sürekli artış göstermektedir. Ayrıca, her kanser hastalığı her hastada aynı şekilde seyretmemekte ve her hasta aynı tedaviye aynı yanıtı vermemektedir. Bu aşamada, bireysel tıp stratejilerinden da yararlanarak daha doğru prognoz tahminleri yaparak daha etkili tedaviler uygulamak oldukça önemlidir. Kanser hastalarında olay zamanı değişkenlerinden yararlanarak sağkalım tahminlemesi yapmak bize çok önemli sonuçlar sağlamaktadır. Omics teknolojilerinin de gelişmesiyle birlikte artık sağkalım zamanı ve hastaların gen ifade profilleri arasındaki ilişki modellenebilmektedir. Bu çalışmalarda son yıllarda avantajlarından dolayı RNA-dizileme verileri kullanılmaktadır. Ancak, RNA-dizileme verileri klasik sağkalım verilerinden farklı olarak yüksek-boyutluluk, heterojenlik ve yüksek-korelasyonlı genleri bulundurma özelliklerine sahiptir. Bu özelliklerinden dolayı klasik sağkalım algoritmaları yerine düzenlileştirilmiş Cox yöntemleri ve sağkalım verilerine uyarlanmış makine öğrenmesi algoritmaları kullanılmaktadır. Ancak, düzenlileştirilmiş Cox yöntemleri Cox algoritmasının kullanımında sağlanması gereken bir takım varsayımları gerektirmektedir. Genellikle önce sınıflandırma problemleri için oluşturulup daha sonra sağkalım verilerine uyarlanan makine öğrenmesi algoritmaları da ek bir zaman ve çaba gerektirmektedir. Bu çalışmada, voom dönüşümü, stacking algoritması ve bloklu lasso yöntemlerini birleştirerek RNA-dizileme verilerinin sağkalım analizlerinde kullanılabilecek yeni yaklaşımlar geliştirilmesi amaçlanmıştır. Bu amaçla, stacking algoritması ile sağkalım verileri ikili sınıflandırma verilerine dönüştürülebilmektedir. Voom dönüşümü sonrası elde edilen gözlem ağırlıkları da priority-Lasso ve IPF-Lasso algoritmalarında kullanılarak voomStackPrio ve voomStackIPF adında iki adet yeni yaklaşım sunulmuştur. Geliştirdiğimiz bu yaklaşımlar TCGA veritabanından alınan on iki adet gerçek RNA-dizileme verisinde uygulanmıştır. Harrell'ın Concordance İndeksi kullanılarak literatürde yer alan diğer sağkalım algoritmaları ile performans karşılaştırılması yapılmıştır. Sonuçlar, çalışma kapsamında geliştirilen iki adet yeni yaklaşımın performansının diğer sağkalım algoritmaları ile benzer veya daha iyi olduğunu göstermiştir.

Anahtar Kelimeler: sağkalım, RNA-dizileme, voom, stacking, IPF-Lasso

INDEX

SYMBOLS and ABBREVIATIONS

AdaBoost	Adaptative Boosting Algorithm
ACC	Adrenocortical Carcinoma
AFT	Accelerated Failure Time
CESC	Cervical Squamous Cell Carcinoma and Endocervical Adenocarcinoma
CNN	Convolutional Neural Network
CPM	Counts Per Million
CPU	Computer's Central Processing Unit
DE	Differential Expression
EN	Elastic-net
ESCA	Esophageal Carcinoma
FPKM	Fragments per kilobase per million mapped fragments
GBM	Glioblastoma Multiforme
GC	Guanine-Cytosine
IPF-Lasso	Integrative Penalized Regression with Penalty Factors
KIRC	Kidney Renal Clear Cell Carcinoma
KIRP	Kidney Renal Papillary Cell Carcinoma
LAML	Acute Myeloid Leukemia
Lasso	Least Absolute Shrinkage and Selection Operator
LGG	Brain Lower Grade Glioma
MESO	Mesothelioma

MDL	Minimum Description Length
NGS	Next Generation Sequencing
PAAD	Pancreatic Adenocarcinoma
PH	Proportional Hazards
RBF	Radial Basis Function
RLE	Relative Log Expression
rlog	Regularized Logarithm
RPKM	Reads per Kilobase per Million Mapped Reads
RNA-seq	RNA-sequencing
ROS	Random Oversampling
RUS	Random Undersampling
SARC	Sarcoma
SMOTE	Synthetic Minority Sampling Technique
SVM	Support Vector Machine
voom	Variance Modeling at the Observation Level
voomStackIPF	voom-based IPF-Lasso
voomStackPrio	voom-based Priority-Lasso
VST	Variance Stabilizing Transformation
TMM	Trimmed Mean of M-values
TPM	Transcripts per million
UVM	Uveal Melanoma
XGBoost	Extreme Gradient Boost

FIGURES INDEX

TABLES INDEX

1. INTRODUCTION

1.1. Problem Overview

Gene expression profiling measures the actively expressed genes in a cell at a specified time. This method produces patterns of genes expressed by a cell, utilizing the capability to simultaneously measure the expression level of transcripts (mRNA or miRNA) for thousands of genes (1). Gene expression profiling has many goals: (i) it evaluates gene activity in particular cell behaviors (e.g., cell division) to determine the cell's role in these processes (2), (ii) it identifies active genes that respond to changes in the cell's environment, improving our understanding of how different conditions affect gene expression, (iii) it studies the role of molecules such as drugs on cell response, and explores potential treatment options by targeting genes that are more prominent in diseases like cancer (1,3).

Transcriptomics technologies are pivotal in the extrapolation and analysis of gene expression. The main technologies used are DNA microarrays and RNA-sequencing (RNA-seq). They identify and quantify gene activity (expression) for gene expression profiling (4). While both methods can detect RNA transcripts in a sample (cells, tissues, etc.), the methods used are distinct. While RNA-seq uses a sequencing approach, microarray uses a hybridization approach. For decades, microarray technology has been used extensively in gene expression research. However, there are limitations to this array technology (5). For instance, the dynamic range for detecting transcript levels in microarrays is somewhat limited, influenced by factors like background, saturation, spot density, and quality, especially when dealing with transcripts found in low abundance (6). In addition, in microarray analyses, cross-hybridization results in high background levels, and microarray techniques rely on *a priori* knowledge of the reference genome (7,8). Due to its numerous advantages, RNA-seq technology, utilizing next-generation sequencing (NGS), has recently become the preferred choice over microarrays. RNA-seq has a wider dynamic range of expression levels and relatively higher sensitivity, allowing the detection and quantification of both highly expressed and low-expressed genes, and contains a very low background signal (5). RNA-seq can identify rare transcripts and low-abundance RNA molecules such as single nucleotide polymorphisms (SNP) except *de novo* SNPs

for low abundance RNAs, while microarrays cannot detect SNPs. RNA-Seq not only identifies transcripts corresponding to known genomic sequences but also sequences complex transcriptomes and explores non-model organisms with undetermined genomic sequences—capabilities beyond the reach of microarrays, which are limited to known sequences. The accuracy of the RNA-seq in detecting the expression of extremely abundant genes is high, while the accuracy of the microarray data is relatively low (9). Finally, RNA-seq eliminates the necessity for specific probes and can sequence without relying on a reference genome (5).

Three main types of gene expression studies can be distinguished: class prediction, class discovery, and differential expression analysis (DE). DE studies are goal-oriented. Identification of genes that express differently under various experimental conditions is known as differential expression analysis (10). In order to treat a gene as differentially expression, the number of reads (or expression levels) between these conditions has to be statistically significant. Class discovery is the process of classifying data by similarity in behavior or property. This means you can discover new classes without using pre-defined labels (11). Class prediction, on the other hand, entails the development of decision rules to discriminate samples with known class labels and determine the class to which a new sample belongs (11). In this thesis, 'survival analysis' of gene expression data will be concentrated, in contrast to these three. This thesis aims to bring a new perspective to the literature on the survival analysis. Predicting when and with what probability a new sample will experience a specific event using gene expression data holds great importance, especially in bioinformatics.

Prediction of survival, especially in cancer patients, is an important factor in clinical decision-making (i.e., increasing the frequency of follow-up and prescribing specific treatments) for clinicians (12). Survival analyses are employed to identify disease-causing factors in cancer patients, estimate the time until death, and predict the degree of malignancy and the time of disease progression. Early detection of cancer and timely appropriate interventions help prevent over-treatment (e.g., unnecessary drug use) and ensure appropriate palliative care is provided to the patient.

Predicting survival is difficult due to the heterogeneous structure of cancer cells. As an illustration, individuals with diffuse large B-cell lymphoma (DLBCL)

demonstrate clinical diversity; while 40% respond favorably to treatment and experience extended survival, the remaining 60% show resistance and have reduced survival prospects. Further investigation revealed significant differences in DLBCL survival rates between activated and germinal center B-like DLBCL (13). In DLBCL, individuals suffering from germinal center B-like demonstrated markedly better overall survival compared to those with activated B-like. In initial cancer survival studies, clinical variables such as age, race, and laboratory results (tumor size, tumor grade, etc.) served as predictors of outcome (14). Nevertheless, depending solely on clinical variables, laboratory results, and clinician experience has proven insufficient in predicting cancer survival (15) because each type of cancer progresses uniquely in each patient, and individual responses to identical treatments can vary significantly. Survival analysis using gene expression data became feasible with the development of omics technologies and precision medicine (16). It has been observed that more accurate results are achieved when utilizing genetic data alone or in combination with clinical data (17,18).

Unlike regression analysis and other classification methods, survival analysis is focused on the time to event outcome. The survival analysis aims to predict survival/hazard functions, compare those functions, and identify the relationship between survival time and covariates. Binary classification techniques or logistic regression, on the other hand, are techniques for predicting the probability of a binary outcome (e.g., smoking status). In a survival analysis, the variables and time of an event are combined to determine its outcome. Certain observations might be censored as a result of this outcome variable. Individuals still alive at the end of the study or lost to follow-up during the study period are considered censored since their survival times are not precisely known (19). Various statistical models exist to predict survival probability in cancer studies. The survival curves between the two groups were compared, and survival functions were estimated using non-parametric techniques like Kaplan-Meier (20), the log-rank test (21), Nelson-Aalen (22), and life-table (23). To account for their effects, semi-parametric models like Cox regression (24) emerged because these non-parametric methods do not provide information about the contribution of different risk factors to the probability of survival. Several Cox regression models have been developed, including penalized Cox regression (25,26),

time-dependent Cox models (27), etc. Cox models require some assumptions, such as proportional hazards, which are often violated in real-life data. Parametric models have been developed that do not require satisfying these assumptions, such as Buckley-James linear regression (28), penalized regression (29), accelerated failure time models (30), etc.

Gene expression data, in contrast to traditional clinical data, suffers from a high-dimensionality issue because there are substantially more genomic covariates ($p >> n$) than samples. The high-dimensionality issue prevents the aforementioned survival analysis techniques from being used with gene expression data such as RNA-seq (31). Gene correlations in RNA-seq are frequently very strong, which can cause serious problems with collinearity (32). The high multicollinearity and high number of genes in the RNA-seq data, in spite of the low sample count, lead to overfitting issues. Moreover, the data structure of RNA-seq is heterogeneous and complex. Although the Cox regression model, a linear model producing risk scores based on covariates, is widely used in survival analysis, it struggles to analyze complex nonlinear relationships between logarithmic risk scores and covariates (33). To address these challenges, penalized Cox regression algorithms, such as lasso (34), ridge (35), and elastic-net (36), perform variable selection on gene expression data (37). Although these algorithms reduce computational costs and overfitting, their application is restricted by the assumptions required for the Cox proportional hazards model. Consequently, recent studies have adopted machine learning approaches for survival analysis, offering more effective solutions independent of model assumptions. Machine learning methods designed for classification problems are powerful and robust. Due to the time-to-event outcome variable, these algorithms cannot be directly applied to survival data. However, upon examining machine learning algorithms developed for survival problems, it is generally observed that they are extensions of machine learning methods originally designed for classification problems, adapted to handle survival data. For instance, random forest algorithms (38), commonly used in classification, have been adapted for survival data, resulting in random survival forest algorithms (39). Other machine learning algorithms follow a similar process, including survival trees (40,41), Bayesian networks (42), neural networks (43), support vector machines (44), ensemble methods

(39,45,46), deep neural networks (47), active learning, transfer learning (48), multi-task learning (49).

Survival algorithms employed in analyzing high-dimensional data have been compared in multiple studies within the literature. For example, Bovelstad et al. (50) evaluated the performance of the following techniques on high-dimensional datasets: univariate feature selection, principal components regression, forward stepwise selection, lasso and ridge regression, and partial least squares regression. Similarly, Van Wieringen et al. (51) assessed the results of numerous survival analysis approaches on high-dimensional genomic data, including univariate Cox regression, principal component analysis, tree-based ensemble methods, penalized least squares, and penalized Cox regression. Moreover, Witten et al. (31) investigated various strategies for survival data analysis of genomic datasets, categorizing them into discrete feature selection, shrinkage-based methods, clustering-based methods, and variance-based methods. A recent study by Spooner et al., (52) used two datasets to test a variety of machine learning (ML) and feature selection algorithms, such as penalized, boosting and random forest methods, for survival analysis. Herrmann et al. (53) performed a large-scale comparison study of multi-omics data to survival. They used eleven survival methods groups: boosting, penalized regression, and random forest.

In some cases, penalized or machine-learning approaches are used to analyze RNA-seq data. The Random Survival Forest algorithm was used in the Ma et al. (54) study on lung adenocarcinoma (LUAD) to analyze RNA-seq and clinical data. The results showed that the RSF model performed better than the classical Cox model. Different deep learning models were used to predict survival in cancer patients using RNA-seq dataset in a study by Huang et al. (55). Ching et al. (56) utilized Cox regression with neural networks (Cox-nnet) to forecast the survival of RNA-seq data. Wang et al. (57) used RNA-seq data to develop a new method for predicting lung cancer survival using a deep learning model based on a Convolutional Neural Network (CNN). Grimes et al. (58) demonstrated that survival analysis results using RNA-seq data gave higher accuracy than those based solely on clinical data, with the elastic-net algorithm delivering the best performance. Jardillier et al. (59) compared the performances of lasso-based penalized Cox Methods (lasso, ridge, elastic-net, adaptive

elastic net, etc.) using 16 cancer datasets. Compared to models established using only clinical data, this study demonstrated enhanced performance when integrating RNA-seq data with clinical data. Ding et al. (60) developed a machine-learning survival prediction model based on miRNAs.

The number of machine learning algorithms developed for classification problems is continuously increasing. Machine-learning algorithms for survival analysis are generally derived from those utilized in the classification problems described above. Applying machine learning methods originally designed for classification problems to survival data requires additional effort. It is crucial to emphasize that all machine learning algorithms developed or yet to be created for classification problems can be applied similarly to address survival problems. Therefore, the idea of stacking becomes significant, converting survival data into classification data and ensuring the use of classification algorithms in survival analysis (61). The stacking changes the data structure, transforming the time until the event outcome in the survival data into a binary outcome variable suitable for classification algorithms. Covariates of RNA-seq gene expression data are presented in two sets: a covariate matrix consisting of continuous variables and a risk matrix consisting of binary variables. While stacking has proven successful with low-dimensional data, no studies in the literature explore applying this stacking concept to high-dimensional RNA-seq survival data (61).

In an RNA-seq experiment, cDNA fragments are assembled by adding sequencing adapters, creating a library of cDNA fragments. Then, this library is sequenced to generate millions of short sequence reads corresponding to individual cDNA fragments (62). Therefore, RNA-seq technology yields data in the form of count numbers. Due to the count nature of the data, analyses of RNA-seq data have employed either discrete distributions such as Poisson (63) or negative Binomial (64,65). However, RNA-seq data has problems with mean-variance dependence, outliers, and high skewness (66). Therefore, since modeling them using count distributions is difficult and complex, studies employing transformation methods have also been applied to apply normal-based approaches by converting discrete count data into continuous data. The RNA-seq dataset in these studies was transformed using logarithmic transformation (67), variance-stabilizing transformation (VST) (65),

regularized logarithm (rlog) transformation (68), and variance modeling at the observation level (voom) (69), all of which were based on normal-based statistical methods. With the voom transformation, the relationship between mean and variance is taken into consideration when modeling discrete count data using a linear modeling technique. This method generates logCPM values for each count data and weights based on observational/sample. However, in most studies utilizing the voom method to analyze RNA-seq data, only the logCPM values obtained are used, and the weights created by the voom method are ignored. Very few studies include logCPM values and weights obtained after voom transformation in the analysis. These studies achieved high accuracy results in differential analysis (70), classification (71), and clustering (72) by utilizing logCPM values and weight values obtained through the voom method with RNA-seq data. However, no survival analysis study was found in the literature that explores the joint utilization of the two outputs (logCPM and weights) obtained after the voom transformation on RNA-seq data.

The stacking algorithm and voom transformation are used to generate the covariate matrix based on the RNA- seq survival data. The covariate matrix is composed of continuous as well as binary variables. In these cases, rather than using traditional machine learning techniques for analysis, we have found that using machine learning approaches that are able to handle different types of data will yield more precise results. The priority-Lasso and Integrative-Penalized Regression with Penalty Factors (IPF-Lasso) algorithms can analyze variables of different data types within distinct blocks. The priority-Lasso algorithm typically organizes variables into blocks based on their types and analyzes them in a prioritized sequence (73). On the other hand, the IPF-Lasso algorithm analyzes diverse data types by assigning distinct penalty factors to reduce the coefficients (74). Both algorithms can also sample weights in their analyses. Employing sample-based weights derived from the voom transformation in these algorithms is likely to yield more precise results. No study in the literature yet performs survival analysis by taking into account the sample weights in the priority-Lasso and IPF-Lasso algorithms on RNA-seq.

This study aims to transform RNA-seq survival data into classification data by combining the powerful voom transform and stacking idea and generate two novel

approaches of priority-lasso and IPF-lasso algorithms, which are adept at analyzing data within a block structure.

1.2. Contribution

This thesis presents two novel algorithms, voom-based priority-Lasso (voomStackPrio) and voom-based IPF-Lasso (voomStackIPF). These algorithms integrate three powerful methods –voom transformation, stacking idea, and lasso with block– for the survival analysis on RNA-seq data. In both approaches, the process is started by transforming raw RNA-seq data with voom. The idea of stacking is then applied to the resulting data matrix. Finally, priority-Lasso and IPF-Lasso algorithms are run using the sample weights derived from the stacked data matrix and the voom transformation. The resulting linear estimators are then utilized to make survival predictions. Thus, the main objectives of proposing these approaches are as follows:

1. to extend the application of voom transformation for survival analysis on RNA-seq data,
2. to adapt the stacking idea for RNA-seq data,
3. to make the priority-Lasso and IPF-Lasso algorithms available for RNA-seq data with sample weights.

1.3. Organization of This Thesis

The organization of this thesis is as follows. The 'General Information' section discussed survival analysis and RNA-seq technology. It was mentioned which methods are used in filtering, normalization, and transformation, leading to the preparation of RNA-seq data for analysis. The section also elucidates the fundamental concept of survival analysis and outlines the algorithms used for analyzing high-dimensional RNA-seq data In the 'Material and Methods' section, we explain how voomStackPrio and voomStackIPF algorithms were created. We also explain how these algorithms are evaluated, and compare them with other algorithms based on real data. We also provide information on the R package we developed for this study. The 'Results' section presents the analysis results from 12 real datasets. The study's findings are summarized in the 'Conclusion' section, while the 'Discussion' section delves into a comprehensive exploration and interpretation of the results.

2. GENERAL INFORMATION

2.1. Cancer and Survival Analysis

In 2020, nearly 10 million lives were claimed by cancer, making it the world's second leading cause of death, which equates to almost one in six deaths (75). Additionally, the 2022 report from the World Health Organization highlights the prevalence of certain cancers in 2020, including breast with 2.26 million cases, lung with 2.21 million cases, colon and rectum with 1.93 million cases, prostate with 1.41 million cases, non-melanoma skin with 1.20 million cases, and stomach with 1.09 million cases. In 2020, the leading causes of cancer-related deaths were lung (1 point 80 million deaths), stomach (769,000 deaths), liver (830,000 deaths), colon and rectum (916,000 deaths), and breast (685,000 deaths). Approximately 400,000 cases of cancer are diagnosed in children each year. Cervical cancer is predominant in 23 countries, with varying prevalence rates across each nation.

Early detection is critical to successful cancer treatment. If treatment is delayed, it reduces the patient’s chances of survival, exacerbates treatment complications, reduces quality of life, and increases treatment expenses (76). Most types of cancer can be detected early. When cancer is caught early and treated early, the patient's five-year survival rate is significantly higher than when diagnosed later. Illustratively, the National Cancer Institute's data indicates that when cervical cancer is detected at an early stage, there is a 92% 5-year relative survival rate. On the other hand, survival rate over five years is 17% in cases of cervical cancer that is discovered after it has spread throughout the body. In addition to significantly impacting survival rates, early diagnosis results in significant cost savings. Early cancer diagnosis has been shown to save the US economy $26 billion annually, according to a study (77).

For the reasons mentioned above, developing new treatment methods is very important for early diagnosis of diseases like cancer, patient survival prediction, and overall survival extension. Utilizing patient data specific to a disease enables the prediction of an individual's likelihood of recovery, mortality, or the probability and timing of mortality through statistical analysis. The time-to-event of interest is a commonly used outcome variable in cancer studies. Different statistical analyses are applied when the time-to-event of interest is observed for all samples. Some samples,

though, might not have experienced the event by the end of the study in some studies, like those on cancer. For these samples, the time until the event is unknown and is classified as ‘censored’. Ignoring these censored samples can result in biased and inefficient estimates (78). Survival analysis is necessary for datasets of this nature. The main goals of survival analysis are to estimate and compare survival/hazard functions and evaluate the relationship between explanatory variables and survival time (79).

Survival analysis stands as one of the most prevalent statistical techniques employed to assess a patient's mortality risk and identify prognostic factors influencing that risk. These analyses aim to estimate life expectancy by observing individuals with a certain disease for a certain period of time, determining the types of treatment, and examining the recovery period or relapse period after treatment. Additionally, survival analyses are valuable in evaluating the impact of newly produced drugs or a newly developed treatment method on patients. It allows comparison of the life expectancy of different patient groups and helps determine whether a disease seen in different regions and times has epidemic characteristics.

Survival analyses are essential for modeling a variety of biological events, including the time from birth to death, the time from cancer treatment to death, the time from the first heart attack to the second, and the time to tumor recurrence. These analyses further calculate probabilities such as 2-year survival, 5-year survival, disease-free survival, progression-free survival, or overall survival. There are several reasons why accurate survival probabilities are important. Over-estimating a patient’s survival may result in delayed treatment for patients with severe disease, allowing them to progress further. On the other hand, under-estimating can lead to patients delaying treatment because they don’t expect to live long enough to see the long-term benefits. Accurate forecasts also help patients and their families deal with life-altering events, allowing them to plan for the rest of their lives accordingly. In addition, precise survival estimates play an important role in the efficient use of scarce healthcare resources by avoiding unnecessary medication and treatment (80).

In medical applications, survival analyses are essential because they can predict the prognosis of a disease and, based on those predictions, estimate the probability that a patient will recover. Survival models provide answers to questions like, “How likely is the patient to survive in 6 years based on the patient’s information?”. These

estimated probabilities are used by clinicians to make important decisions about patient care. For example, they may increase the frequency of follow-ups or perform specific treatments. Accurate prognosis predictions help clinicians make appropriate clinical decisions in treatment and care planning and reduce the risk of over- or under-treatment. For example, although mandatory rehydration is routinely performed for cancer patients with fatal diseases, the importance of stopping or withdrawing rehydration is emphasized to avoid distress due to overhydration (81). Similarly, though corticosteroids and sedatives often provide relief for symptoms, their long-term unnecessary use can lead to undesirable effects such as Cushing's appearance, oral candidiasis, and tolerance (82). Consequently, decisions regarding medical applications are largely contingent on survival assessments.

Diagnostic research and application centers are dedicated to investigating the genetic causes of diverse diseases, delving into pharmacogenetics and personalized medicine, and applying genetic tests using survival analysis. In order to ascertain the impact on life expectancy, cancer research centers conduct survival studies on a variety of cancer types and other malignancies. In the meantime, survival analyses are used by biotech and pharmaceutical companies to assess the efficacy of novel drugs. In addition to these applications, researchers from a variety of industries frequently employ survival analyses in their scientific investigations.

2.2. Survival Analysis in Precision Medicine

Traditionally, pathological exams and symptom observation have been the primary methods used by physicians to diagnose cancer. The pathological examination method involves looking at the cancerous cell under a microscope. It has been used for many years to diagnose cancer. But this method, which is based on a variety of criteria and the experience of experts, is by its very nature subjective. Additionally, the challenge arises when different tumors share the same DNA, making accurate diagnosis challenging. In response, analyses utilizing gene expression data play a vital role as the distinct gene expression profiles among various tumors differ (83). Using people’s genetic information to diagnose cancer will result in faster, more precise, and more sensitive findings.

The primary modalities employed in cancer treatment include surgery, chemotherapy, and radiation. Additionally, supplementary approaches include targeted therapy, immunotherapy, laser treatment, and hormonal therapy. But even the same cancer can progress differently in different patients. There are also differences in cancer types and types of cancerous cells. These factors limit the effectiveness of traditional cancer treatments. Consequently, researchers are increasingly looking for personalized approaches in cancer treatment. Precision medicine, alternatively referred to as personalized medicine, encapsulates the idea of administering the right drug to the right patient at the right dose and time (84). Advancements in high-dimensional sequencing technology have made obtaining genetic data, including genomic, transcriptomic, metabolomic, etc., more accessible. By using this genetic information, precision medicine can identify high-risk patients before symptoms appear, highlighting the critical role that early detection plays in the diagnosis of many cancers. Early diagnosis improves patient survival, maximizes financial resources for healthcare, and lowers the risk of severe conditions. As such, precision medicine, in its goal of preventing the disease process, minimizes or eliminates side effects during treatment, allowing patients to derive maximum benefit from the therapy and achieving the ultimate purpose of disease treatment (85).

In classical survival analysis, patient demographic and clinical data are commonly employed, resulting in the calculation of similar survival probabilities for individuals sharing similar demographic and clinical characteristics. Now that personalized medicine applications are available, we've noticed that survival time and survival chances can differ greatly from person to person. Distinct molecular and patient characteristics can lead to diverse progressions of the same disease, and individuals may exhibit different responses to identical treatments (Figure 2.1). Despite the physical similarities, the differences in responses are mainly due to genetic differences, which is why genetic information is such an important part of precision medicine. Precision medicine recognizes that even individuals with the same genetic origin may experience distinct progressions of fatal diseases. Consequently, predicting survival times using specific biomarkers related to the prognosis of such diseases becomes essential. Various biomarkers utilized in personalized treatments for predicting survival, identifying high-risk groups, and forecasting benefits from

specific treatments have significantly contributed to the diagnosis and prognosis of diseases in clinical studies (86). Consequently, alongside classical survival analyses utilizing clinical data, recent advancements have led to the development of survival models incorporating high-dimensional molecular data from technologies such as omics (genomics, transcriptomics, proteomics, metabolomics).

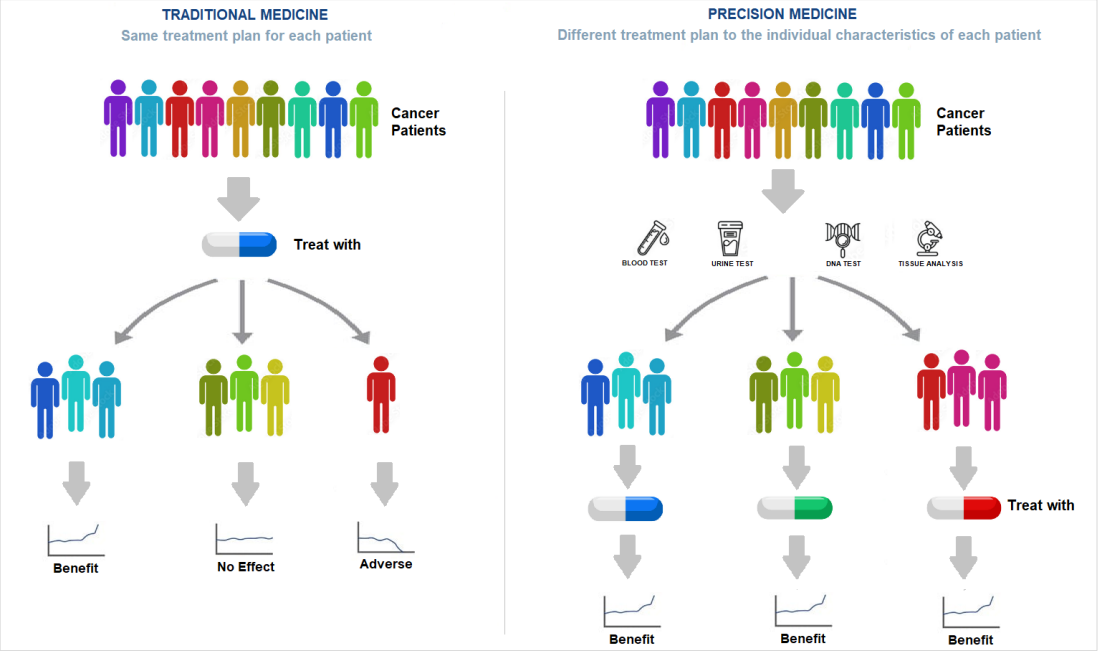


Figure 2.1. Differences in treatment processes and outcomes between traditional and precision medicine.

Sequencing technologies are instrumental in establishing the genetic profiles of tumors in cancer patients. Gene expression data serves as a snapshot of the diseased gene, and the intensity of expression of specific genes within diseased tissue is a good biomarker for predicting the probability of patient survival. There is a high correlation between gene expression data and survival, and several studies have shown that the power of such data is more remarkable than clinical data and other prognostic factors (87). Genes expressing cancer cells can be identified and treatment response of a patient can be predicted through analysis of genetic profiles using sequencing technologies. By using sequencing technologies, cancer patients can now receive more individualized and customized treatment plans based on the unique characteristics of their cancer, as opposed to the standard application of surgery, chemotherapy, and radiation treatments to every patient.

2.3. Next Generation Sequencing Technologies

Next-generation sequencing (NGS) stands as a high-throughput method proficient in sequencing vast and complex genomes, suitable for both DNA and RNA samples. NGS technology is distinguished by its high-speed capabilities. For instance, the process of sequencing the entire human genome, which once took more than a decade with the older Sanger sequencing technology, can now be completed in just a single day using NGS (88).

The way NGS works is similar to that of Sanger sequencing; however, there are some key differences. NGS can detect genomic variations, which is more sensitive and quantitative than Sanger sequencing can. NGS can generate more sequencing data on the same set of input sample requirements that Sanger sequencing does. NGS employs massive parallel sequencing and simultaneously screens multiple genes across multiple samples. It does not require *a priori* knowledge of the genome. NGS is sensitive to tumor heterogeneity, leveraging its capacity to sequence heterogeneous genomes within a sample. Additionally, NGS offers a single-nucleotide resolution. It also has a higher dynamic range of signal, reproducibility and a lower sequencing cost (89).

Various NGS platforms utilize diverse sequencing technologies, enabling the simultaneous sequencing of numerous DNA polymers. Each NGS platforms conducts parallel sequencing of millions of small DNA fragments. Illumina is the most widely adopted among these platforms, and its workflow is illustrated in Figure 2.2.

Sample preparation begins with the extraction of DNA for next-generation sequencing (90). Before employing NGS technology, the sample must undergo the following steps to prepare for sequencing.

Sample Extraction: This step aims to obtain pure DNA or RNA. DNA or RNA nucleic acids are extracted from various biological samples, including blood, cell cultures, sputum, bone marrow, tissue selection, bacterial cultures, or urine (90). Different extraction methods are employed based on the starting material. The purpose of these methods is to obtain the best quality and the highest yield of nucleic acids from the sample type. Nuclear acid isolation involves disrupting the cell wall or cell membrane disruption through physical, chemical, or enzymatic methods to release the genetic material. Subsequently, in nuclear acid isolation, undesirable substances such as

proteins and lipids that may interfere with the reaction are eliminated from the cell using methods like centrifugation, filtration, or bead-based. Following nucleic acid isolation, a purification step is initiated. Various methods, such as silica, ion exchange, cellulose, or precipitation-based techniques, are employed to purify nucleic acids. The final stage involves assessing the amount of nucleic acid in the sample. Inadequate nucleic acid concentration may lead to the amplification of unwanted products during polymerase chain reaction (PCR) or the generation of short-read lengths during sequencing. An increased background during sequencing procedures may result if the nucleic acid concentration is too high.

Library Preparation: This step transforms the extracted nucleic acids into a format suitable for the chosen sequencing technology (90). Generating a sequencing library from a DNA or RNA sample involves two main steps: (i) amplification and (ii) the addition of sequencing adapters. In the case of RNA as the starting template, an additional step is required to convert RNA to cDNA through reverse transcription. First, all DNA is fragmented into similar-sized pieces to enhance the reading sensitivity of the bases and mitigate enzymatic errors associated with longer DNA strands. Various methods, including physical, chemical, or enzymatic approaches, are employed for DNA fragmentation. Once the length of the DNA is adjusted, specialized adapters are ligated to both ends of the DNA fragments. Adapters, chemically synthesized oligonucleotides with predetermined sequences, bind to the ends of DNA molecules. These adapters are designed to interact with a specific sequencing platform and serve as barcodes, enabling the identification of the initial location of each nucleotide.

Clonal Amplification: The DNA fragments from the libraries are amplified to such an extent that fluorescent signals for single-base incorporation are detectable by the sequencers in the downstream sequencing reaction. Initially, the library created from DNA fragments is fixed to the surface for amplification. The fragments are hybridized to the flow cell surface, and each bound fragment is amplified into a clonal cluster through a series of amplification reactions known as bridge amplification (91). The five steps of bridge amplification include: (i) synthesis of the complementary strand of a DNA fragment in the library from the priming oligo of the flow cell, (ii) folding of the complementary strand folds and formation of the double-stranded bridge, (iii)

creation of two single strands by denaturing the double-stranded bridge, (iv) repeating the process of bridge amplification, and (v) generating more clones of double-stranded bridges. Subsequently, each fragment forms a cluster of identical molecules known as clonal clusters, each representing one primary library molecule. The double-stranded clonal bridges are denatured, the reverse strands are removed, and the forward strands persist as clusters for sequencing.

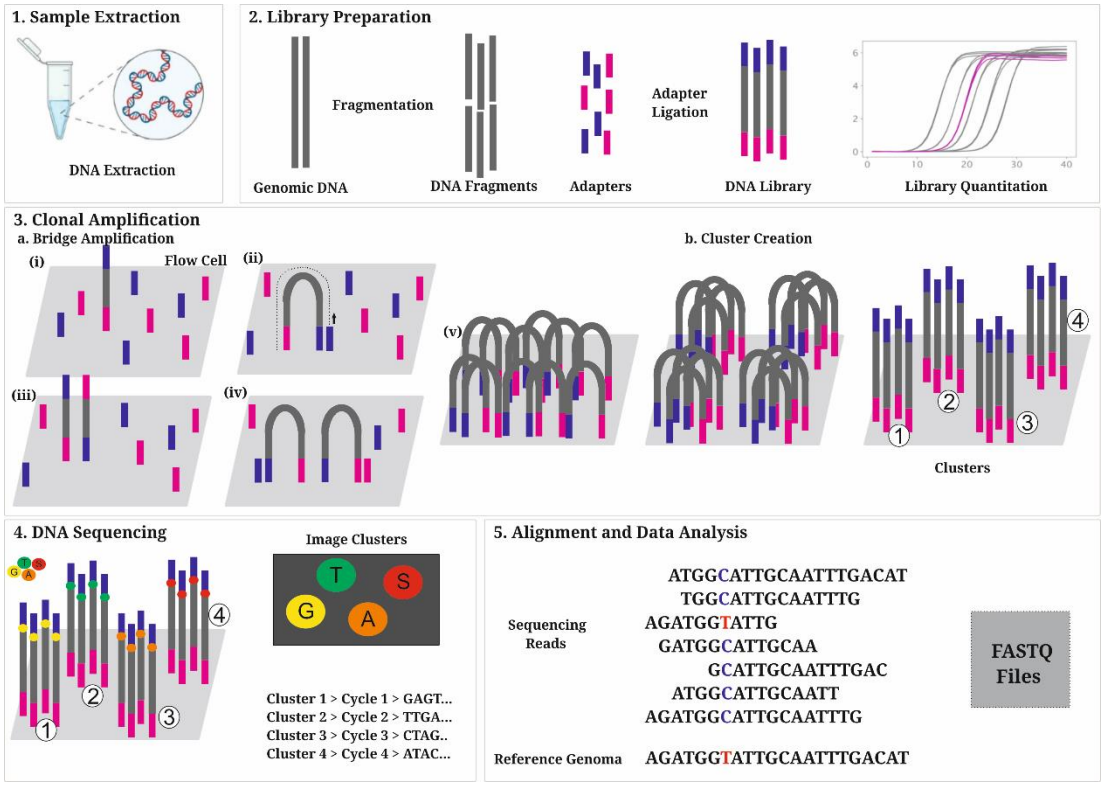


Figure 2.2. Workflow of next generation sequencing using Illumina systems.

DNA Sequencing: NGS platform is used for parallel sequencing. The library is loaded onto the sequencer, in which it systematically 'reads' the nucleotides individually. The DNA sequence obtained by sequencing each piece of DNA is referred to as a read. The quantity of reads generated varies based on the sequencing platform and kit. A comprehensive comparison table of sequencing platforms is provided in Table 2.1 of Zararsiz's PhD thesis (92).

The most popular platform is Illumina sequencing. Illumina sequencing uses fluorescent dye-labeled dNTPs with a reversible terminator to read fluorescent signals in every cycle, using a process called cyclic reversible termination (93). Only one of the four fluorescent DNTPs are incorporated into the DNA polymerase in each cycle

based on complementarity and then unbound DNTP's are eliminated. Cluster images are taken after each nucleotide has been incorporated; the emission wavelength is measured and the fluorescence intensity is measured to determine the base that has been incorporated into each cluster during this cycle. The fluorescent dye and terminator are then cut and released after imaging. This is followed by another synthesis cycle, another imaging cycle, and another deprotection cycle. Because each base is read from one cycle to the next, the read length is iteratively repeated 'n' cycles.
Alignment and Data Analysis: Initially, the reads must be filtered based on quality, amplicon size, and concordance between paired ends. The reads are then assembled and aligned to a reference genome. In the concluding stages, reads can be compared with reference sequences or with other samples to detect variants by disease status, etc. If reads are aligned with a reference genome, variant annotation can associate variants with known genes or regulatory sequences.
This final step comprises three phases: processing, analyzing, and interpreting the raw sequencing data. Various bioinformatics tools, such as TopHat2 (94), STAR (95), featureCounts (96), DESeq2 (68), and EdgeR (97), are employed to process, analyze, interpret, and transform raw sequencing data into meaningful information.

2.4. RNA-Sequencing Technique

A new high-throughput sequencing technique for transcriptome analysis called RNA-sequencing (RNA-seq) offers a reliable method for describing and measuring transcriptomes. Although microarray technology has been used for gene expression profiling studies for many years, RNA-seq offers many advantages over microarray technology. Firstly, unlike DNA microarrays, which can only profile predetermined transcripts/genes, RNA-seq enables comprehensive sequencing of the entire transcriptome (98). Secondly, due to the markedly lower background signals in RNA-seq compared to DNA microarrays, noise in the experiment is easily eliminated during analysis. Third, RNA-seq has a wider dynamic range of expression and does not require a large amount of total RNA for quantification (5). Finally, RNA-seq offers higher resolution, a better detection range, and reduced technical variability (99).

RNA-seq is the direct sequencing of transcripts by NGS. All RNA-seq data is therefore generated using the libraries preparation and sequencing platforms listed in

the ‘Next Generation Sequencing’ section. NGS is capable of generating millions of reads. Depending on the sequencing platform you choose, the number of reads may differ. There are several steps that need to be completed before statistical analysis can be applied to the RNA-seq data and the tools used in these steps are described in detail in Table 2.1.

FASTQ Formats: High-dimensional sequencing results are obtained using next-generation sequencing technologies (such as Illumina) in FASTQ (100) format, often with the .txt extension. This format represents both sequencing data and quality scores using a single ASCII character. Each read is presented with four lines stacked one below the other in the file (Figure 2.3).

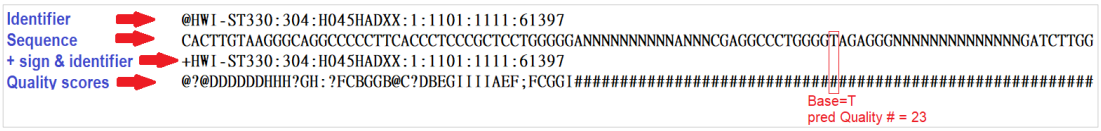


Figure 2.3. FASTQ formats.

In Figure 2.3, Line 1 starts with the @ character and continues with a sequence identifier, typically containing information related to sequencing technology, such as flow cell IDs, lane numbers, and information on read pairs. Line 2 consists of the raw sequence reading featuring sequence letters. Line 3 starts with + and marks the end of the sequence. The sequence identifier on the first line may follow the +. Line 4 displays the quality values corresponding to the sequence in Line 2, containing the same number of symbols as the letters in the sequence.

Quality Control: Quality control involves assessing raw sequencing data to identify potential problems that may affect downstream analyses. Data quality metrics are determined for this, providing information about various aspects such as read length, sequencing depth, base quality, and GC content.

In Figure 2.3, Line 4 of the data in FASTQ format contains the quality code. This code indicates the likelihood of a sequencing error at each nucleotide position, and the quality score is derived when the probability of such an error is known. For instance, if the probability of an ‘x’ error is 0.01, its quality score would be -10*$log_{10}p$=20. These characters on Line 4 of the FASTQ file are interpreted based on the ASCII character table.

The widely used tool for quality metrics is FASTQC (101), which takes the FASTQ file as input and generates an HTML file as output. This HTML file comprises several sections. For instance, the "Basic statistics" section provides general information about the number and length of reads, while the quality of reads of nucleotides is visually presented in the "Per base sequence quality" section.

Filtering/Trimming: Adapter sequences are short oligonucleotides and ligated to DNA fragments' ends. When you read the adapter sequence next to the unidentified target DNA sequence, remove the adapter sequence to restore the target DNA sequence (102). Similarly, low-quality reads containing sequencing errors, such as base-calling errors, phasing errors, and insertion-deletion errors, are excluded from the sequencing data. The use of adapter sequences and low quality nucleotides may result in false positives and lower the accuracy of the downstream analysis.

Sequence Read Alignment: Alignment identifies the optimal position for each read in relation to a reference genome. For organisms that have a reference genome, reads are mapped to a genome or to a transcriptome. Two important mapping quality parameters are the percentage of mapped reads and the uniformity of read coverage on exons and the mapped strand (103). Following alignment, the result file format is Sequencing Alignment Map (SAM) or Binary Alignment Map (BAM).

Expression Quantification: The number of reads from the RNA-seq data that map to each transcript sequence is estimated in this step (103). A gene transfer format (GTF) file is used to count the number of reads that have been mapped or aligned to each gene during the process. GTF files contain gene models illustrating the structure of the transcripts produced by each gene.

De novo Transcriptome Assembly: When an organism's reference genome is either incomplete or nonexistent, *de novo* assembly is utilized. In this step, a reference file is created using the available data because there isn't a reference genome yet.

Following the expression quantification steps, raw count data from RNA-seq is acquired. Now, the pre-processing steps for this raw data have been initiated.

Table 2.1. Tools for steps of RNA-seq data analysis workspace.

Steps for RNA-seq data analysis	**Tools**
Quality Control	FASTQC (101) NGSQC (104) RNA-SeQC (105)
Filtering/Trimming	Trimmomatic (106) PRINSEQ (107) Soapnuke (108)
Read Alignment	Bowtie (109) BWA (110) STAR (95) Tophat2 (94) HISAT2 (111)
De novo Assembly	Cufflinks (112) StringTie (113) Trinity (114) SOAPdenovoTrans (115) Trans-ABySS (116)
Expression Quantification	RSEM (117) Kallisto (118) Salmon (119) FeatureCount (96) HTSeq-count (120) eXpress (121) DEXSeq (122) Sailfish (123)

2.5. RNA-Sequencing Data

2.5.1. Raw Data

The raw RNA-seq data comprises non-negative and integer count data. As illustrated in Table 2.2, the rows represent samples, while the columns represent genes. Time (t) and status (δ) are variables associated with survival in the RNA-seq data. The status variable indicates whether a sample has experienced a specific event. If the sample has experienced the event, the status variable is set to 1. If the sample has not experienced the event (i.e., if it is censored), the status variable takes the value 0. If a sample experienced a specific event, the time variable denotes the duration until the occurrence of that event; if the sample was censored, it denotes the time at which censoring took place.

2.5.2. Filtering

Low-expressed genes in RNA-seq data may negatively affect analysis results. First, RNA-seq inherently contains noise because it is obtained through a natural random sampling process, and accurate expression quantification becomes difficult because it measures gene expression profiles over a wide dynamic range (5). These noise and measurement mistakes are more common in low expression genes in RNA-seq. Secondly, low expression genes are not biologically significant because genes usually need at least a certain amount of expression to turn into proteins or to be considered biologically significant. Thirdly, the mean-variance relationship is more accurately estimated by excluding low-expressed genes from the dataset. Inadequate removal of low-expressed genes adversely affects linear modeling in limma-voom, particularly when working with logCPM values assumed to be normally distributed. Suppose filtering of low-expressed genes is inadequate for linear modeling in limma-voom; the mean-variance trend plot generated as part of the voom function will show a decrease in variance levels at the lower end of the expression scale. Lastly, from a statistical perspective, the sensitivity of detecting differentially expressed genes may be reduced when genes have consistently low-expression counts (124). Hence, identifying and removing low-expressed genes and insufficiently sequenced fragments from each sample's data are biologically and statistically essential.

Numerous methods are available for filtering low-expression genes, such as applying a predefined threshold value (69,97), filtering genes with consistently low-expression across samples, and filtering genes with low variance across samples. The **edgeR** package (97) is commonly employed in studies to filter low-expressed genes. The `filterbyExpr()` function in this package contributes to more accurate analysis by eliminating genes with low-expression from the dataset. This function automatically removes unexpressed or low-expressed genes while retaining as many genes as possible with valuable counts. A gene to be considered expressed in a library, it must have 5-10 counts. By default, this function selects the sample count of the group with the smallest sample count as the minimum sample count and keeps genes with at least ten or more sequence fragment counts in this sample count. The filtering criterion is to remove the gene if the number of genes with less than ten expression counts in all samples exceeds the minimum number of samples (125). This function preserves

genes with counts per million (CPM) greater than *k* in the sample of *n*. Here, *n* is determined by the minimum group sample size, and *k* is determined by the minimum number of samples (dafault:10) at the minimum sample rate (default: 70% of smallest group size). Genes with at least a few counts of 10 or more can be selected, but it is preferable to use CPM values to account for differences in library sizes. For example, if the median library size is 51 million and 10/51 (about 0.2), the function retains genes with a CPM of 0.2 or more in at least three samples. The CPM cutoff used is affected by sequencing depth and the experimental design. A lower CPM cutoff is preferred when library sizes are larger, while a higher CPM is favored in the opposite case. logCPM, FPKM, and RPKM can also be employed as scale conversions instead of CPM (126).

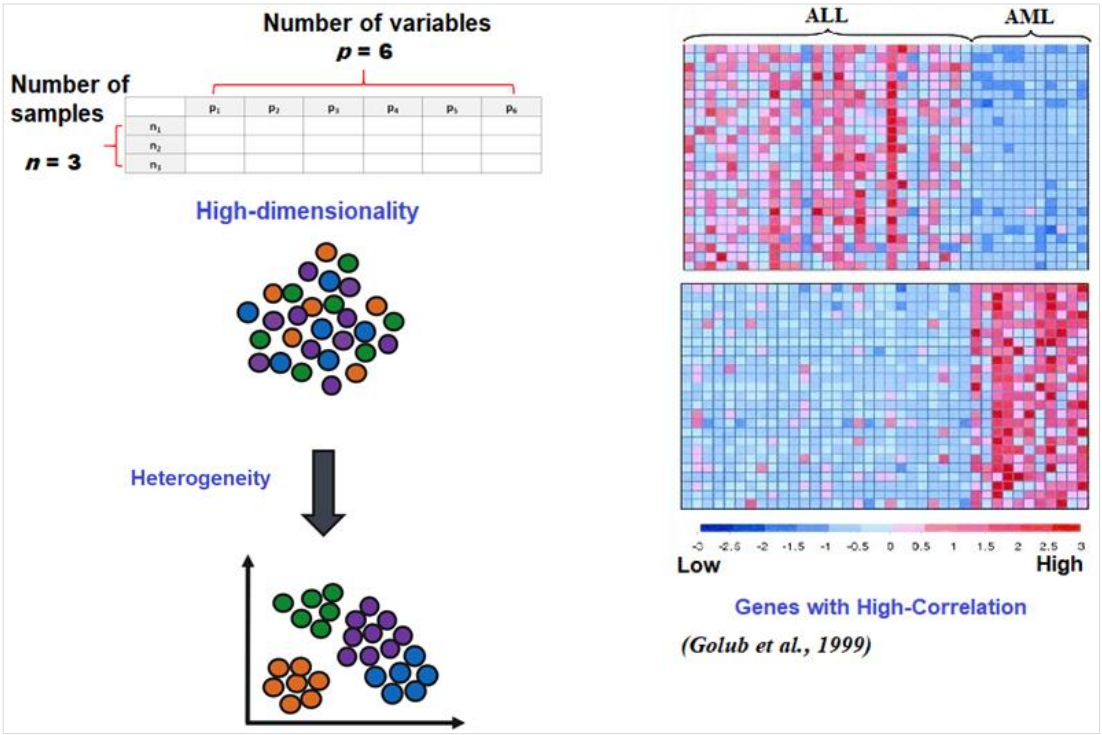


Figure 2.4. High-dimensionality, heterogeneity, and high-collinearity problems of RNA-seq data.

2.5.3. Normalization

Normalization is an essential step in the preprocessing of RNA-seq data prior to analysis. In some cases, there may be technical differences between measurements in different samples or unwanted biological effects such as batch effects (127) or

general noise (128). Normalization methods take into account this sample variability to remove systematic experimental bias as well as technical variations while maintaining biological variation. Technical variations limit comparability as there are differences in measurement distribution between samples. Therefore, the use of standardization algorithms is necessary to eliminate or reduce technical variation.

Table 2.2. An example RNA-seq survival data matrix for ACC data.

Samples	**Genes**								**Time**	**Status**
	Gene1	**Gene2**	**Gene3**	**Gene4**	**Gene5**	**…**	**Gene19930**	**Gene19931**	**(*t*)**	**(δ)**
Sample1	5	100	40	987	8	…	532	6	8	1
Sample2	11	89	6	53	5	…	69	4	17	1
Sample3	6	67	78	61	14	…	74	10	9	0
Sample4	8	69	51	99	9	…	78	19	13	1
…	…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…
Sample79	3	20	12	678	2	…	49	43	20	0

While early RNA-seq studies initially considered normalization unnecessary, subsequent analyses demonstrated its significance (129). A gene's expression level is determined by its number of mapped reads. Normalization is necessary to convert the raw read count into an informative measure of gene expression by addressing factors that affect the number of mapped reads on a gene, such as length (130), GC content (131), and sequencing depth (132). Another reason for the necessity of normalization is the variation in the proportion of mRNA corresponding to a given gene between biological conditions. In the sequenced sample of molecules, the number of molecules (reads) corresponding to a given gene depends on the proportion of that gene in the population of molecules available for sequencing. Therefore, when a few genes are highly expressed in only one of the conditions, these genes will contribute a larger share of the total molecules, leaving a smaller portion of the reads for other genes (132). It may result in inaccurate differential expression for non-differentiated expression genes, which highlights the need for normalization to explain these differences. There are many ways to normalize RNA-seq data.

Total Count Normalization: Each read count is divided by the total number of reads in its corresponding sample to account for differences in library sizes among samples (133).

Upper Quartile Normalization: Initially, genes with zero read counts across all samples are removed. Subsequently, the count for each remaining gene is divided by the 75th percentile (upper quartile) of the counts for its corresponding sample (129).

Median Quartile Normalization: Similar to upper quartile normalization, genes with zero read counts across all samples are removed. However, the count for a remaining gene is divided by the median, rather than the 75th percentile (upper quartile), of the counts for its corresponding sample (133).

Quantile Normalization: It calculates a specific quantile, ensuring uniformity in the distribution of normalized data across all samples by replacing each quantile with the mean (or median) of that quantile calculated across the entire set of samples (134).

Trimmed Mean of M-values (TMM) Normalization: Initially, the TMM (132) approach selects a sample as the reference sample. Subsequently, it compares the counts in each sample to those in the reference sample to estimate the sequencing depths ratio between each sample and the reference. Trim the gene based on fold change and absolute expression level calculated from the selected sample to remove differentially expressing genes. The mean is calculated over genes that do not exhibit differential expression (except for differentially expressing genes). Trim the fold changes by calculating the trimmed mean for each sample and scaling reads counts based on this trimmed mean as well as the number of samples.

Relative Log Expression (RLE)-DESeq Normalization: The DESeq (65) normalization initially computes a ratio, where the numerator is a read count, and the denominator is the geometric mean of all read counts across all samples for that gene. The denominator in this context represents a pseudo-reference sample. This process is applied to every read count. Subsequently, it computes the median of all ratios specific to that sample to scale a sample. This calculated value is the size factor for the corresponding sample. The ratio of size factors calculated for each sample indicates the ratio of their respective sequencing depths.

By computing a virtual pseudo-reference instance, DESeq corrects for overexpressions resulting from gene length and frequency biases. For the dataset with the p gene and n

sample, the pseudo-reference sample is calculated as the geometric mean of counts across all samples for the g^{th} gene, forming a vector of the gene geometric mean ($s_g = \sqrt[n]{r_{g1}r_{g2}\dots r_{gn}} = (\prod_{g=1}^{n} r_{gn})^{1/n}$ *g*=1,2,…,*p)*. The reason for calculating the geometric mean here is that the geometric mean is less sensitive to extreme values than the arithmetic mean.

Subsequently, for each sample, the median of the ratio of the counts of the relevant sample to the pseudo-reference sample is defined as the size factor. For the i^{th} sample, the size factor is calculated as follows.

$$\hat{R}_i = median_g \frac{r_{gi}}{(\prod_{g=1}^{n} r_{gn})^{1/n}} = median_g \frac{r_{gi}}{s_g}$$

The denominator in the equation is an example of a pseudo-reference obtained through geometric mean across samples. Therefore, each estimate of size factor denoted as $\hat{R}_i$, is calculated as the median of the ratios of the counts of the i^{th} sample to those of the pseudo-reference. The size factor is applied to scale that sample (65).

Finally, for each sample *i*, the normalization factor is calculated as the median of the r_{gi} values. Genes with a geometric mean of zero are ignored when calculating this median. DESeq median normalization involves dividing each gene value in each sample by these median values calculated for the relevant sample (135).

$$\hat{r}_{.i} = \frac{r_{.i}}{\hat{R}_i}, i = 1,2,\dots,n$$

PoissonSeq: A group of genes that are non-differentially expressed (non-DE) are first found using the PoissonSeq algorithm (63). It then determines a scaling factor to approximate the read counts expected for every sample. Next, the goodness-of-fit test is used to see if the predicted values agree well with the associated genes. This iterative process is repeated until the algorithm best matches the observed and expected values.

Reads per kilobase per million mapped reads (RPKM), Fragments per kilobase per million mapped fragments (FPKM) Normalization, Transcripts per million (TPM): RPKM normalization (136) involves dividing each read count by the product of the number of reads in the sample (in millions) and the gene length (in kilobases). This

accounts for gene lengths and the library size in total count normalization. FPKM normalization (112), similar to RPKM, divides each read count by the number of reads in the sample and the gene length but uses cDNA molecules instead of RNA reads. TPM (137) is a slight modification of RPKM, converting an RPKM to a TPM using the formula $\mathrm{TPM} = 10^6 * \frac{\mathrm{RPKM}}{\mathrm{Sum(RPKM)}}$.

CuffDiff Normalization (138), an extension of DESeq normalization, is an additional method to the ones mentioned above. Both an internal and an external size factor are computed as two distinct normalization factors. A more resilient version of the TMM approach is the Median Ratio Normalization (139), which is merely an extension of TMM normalization.

2.5.4. Transformation

The RNA-seq count data matrix exhibits sparsity and skewness (140). Sparsity means that many counts in the RNA-seq count matrix are zero. Conversely, skewness refers to a skewed distribution when the histogram is plotted for all counts in the RNA-seq count matrix. Additionally, RNA-seq count matrices are generally heteroskedastic, meaning the number of highly expressed genes varies more than low-expressed genes (141). Analyzing data with unequal variance using standard statistical methods is very difficult. To overcome this problem, various transformation methods can be applied to make the data homoskedastic.

Logarithmic transformation: For data with a skewed distribution, the logarithmic transformation is a simple method that is often employed. This transformation helps the data distribution approximate a normal distribution. When applying a logarithmic transformation to reduce or eliminate the skewness of RNA-seq data, adding a small constant, such as 0.5 or 1, to each count is common. This addition is required to prevent undefinable outcomes in the log transformation, particularly when working with dataset counts that are equal to 0.

$$x'_{ig} = \log(x_{ig} + 0.5) \; or \; x'_{ig} = \log(x_{ig} + 1)$$

After applying the log transform to RNA-seq data, the distribution typically doesn't become perfectly normal, but it exhibits reduced skewness and fewer extreme values.

Variance stabilizing transformation (VST): The purpose of the vst transformation is to obtain variables whose variances independent of the mean, thereby eliminating the dependence of the variance on the mean. This helps prevent a high variance of the logarithm of the count data, especially when the mean is low. This transformation method models the relationship between means and variances with a dispersion parameter (65). Assume that μ_g is the mean and σ_g^2 is the variance for the g^{th} gene. When the relationship between the mean and the variance is modeled by $\sigma_g^2 = \mu_g + \alpha_g \mu_g^2$ and a dispersion parameter is $\alpha_g = \alpha_0 + \alpha_1/\mu_g$, then vst transformation is calculated follows.

$$x'_{ig} = \int_0^{x_{ig}} \frac{1}{\sigma_g^2} d\mu_g$$

The parameters α_1 and α_0 are estimated using generalized linear models. Following the vst transformation, all genes exhibit unequal variances, yet the counts are less skewed and show fewer extreme values.

Regularized logarithm transformation (rlog): The vst transformation may not perform optimally with datasets featuring unequal library size (68). To address this issue, the rlog transformation is introduced. Like the vst transformation, the rlog transformation aims to eliminate the variance dependency on the mean (68). Although many aspects of rlog transformation resemble those of the vst transformation, the rlog transformation requires more time, especially in datasets with numerous samples. This increased time is due to the rlog fitting a shrinkage term for each sample and each gene. The rlog transformation is applied as follows.

$$x'_{ig} = log_2(q_{gi}) = \beta_{g0} + \beta_{ig}$$

The parameter q_{gi} is proportionate to the expected accurate concentration of fragments for the g gene and i sample. The intercept β_{g0} does not undergo shrinkage and β_{ig} is the sample-specific effect which is shrunk toward zero based on the dispersion-mean trend over the entire dataset.

Power transformation: As RNA-seq data comprises non-negative counts, modeling them with a discrete Poisson distribution is appropriate (9,129). Because of these data

with biological replicates, the overdispersion issue arises due to the variance being much larger than the mean. However, it cannot cope with this overdispersed problem since the mean and variance of the Poisson distribution have the same parameter value. An alternative is using the negative Binomial distribution to model RNA-seq data instead of the Poisson distribution (65,132). Nevertheless, due to the complexity of the negative Binomial distribution, Witten et al. proposed applying a power transformation to the RNA-seq data in their study. Although the transformed data is not an integer type after the power transformation, it can still be modeled using the Poisson distribution (63).

When $\alpha \in (0,1]$, the transformed count values are utilized ($x'_{ig} = x_{ig}^{\alpha}$). Using the total count size factor estimation, a test is conducted to assess whether the Poisson model fits the data well, as expressed by the following formula (142).

$$\sum_{g=1}^{p}\sum_{i=1}^{n}\frac{(x'_{ig} - \frac{x'_{.g}x'_{i.}}{z_{..}})}{(\frac{x'_{.g}x'_{i.}}{x'_{..}})} \approx (p-1)(n-1)$$

voom transformation: RNA-seq quantifies the number of sequence reads mapped to each gene or other genomic feature (exons, transcripts, etc.), resulting in RNA-seq datasets consisting of integer counts (65). Consequently, statistical analyses of such data have been approached through methods that analyze log counts after normalization by sequencing depth (143–145) or by modeling using discrete data distributions such as negative Binomial (65,97,146) and Poisson (147). However, the mathematical theory of discrete distributions is less tractable than normal distribution approaches and presents more limitations. Most discrete distribution methods applied to RNA-seq data yield accurate results for datasets with small sample sizes. Additionally, methods based on these distributions are statistical tests treating estimated distributions as known parameters. Commonly used normal-distribution methods for microarray data analysis are unsuitable for RNA-seq read counts because RNA-seq data consists of integer counts, unlike the continuous data format of microarrays. Despite log transformation, RNA-seq data retains the issue of unequal variance—larger counts with larger standard deviations and smaller counts with smaller standard deviations. In order to overcome these difficulties, the voom

transformation was designed to model the mean and variance relationship without specifying the precise probabilistic distribution of counts (69). By incorporating the mean-variance trend into precision weights for each normalized observation, the voom transformation enables the application of statistical methods based on normal distribution after predicting the mean-variance trend of the data.

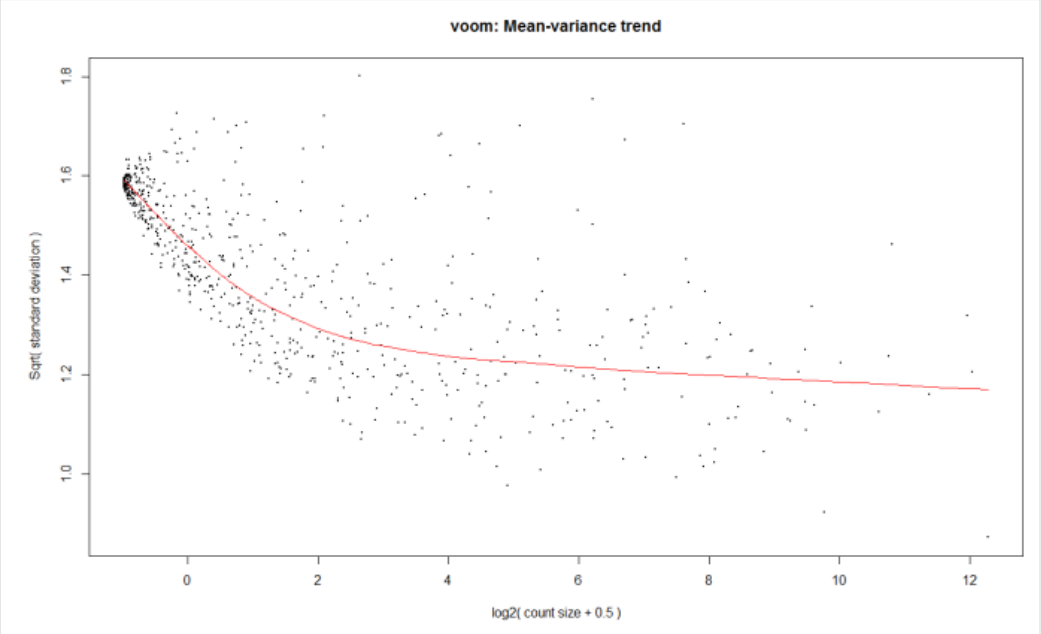


Figure 2.5. Voom mean-variance modeling.

If the RNA-seq datasets comprise n samples, each sample's count of reads matching with each gene defines the RNA-seq profile. These profiles often involve tens of thousands of genes, with the number of samples typically limited. The total number of matched reads for each sample, referred to as the library size, may range from a few hundred thousand to hundreds of millions. The count of reads for a gene is proportional to the gene expression level, the gene transcript length, and the sequencing depth of the library. Counts per million (cpm) values are derived by dividing each read count by library size, enabling comparison across libraries of varying sizes in millions. The differences in logCPM between samples generate the log-fold-changes of the expression. logCPM (logarithm of counts per million reads) values, akin to log-intensity values in microarrays, were utilized; however, it's important to note that logCPM values may not exhibit constant variance.

When analyzing the probability distributions of counts, it is seen that larger counts exhibit larger variances. So, it was observed that the coefficient of variation in RNA-seq should be a decreasing function for small counts and asymptote to a value dependent on biological variability for larger counts (148). Studies involving technical replications have demonstrated the standard deviation of logCPM continuously decreases as a function of the mean. Conversely, in the case of biological replications, this decrease is earlier and relatively more asymptotic (Figure 2.5) (69). Consequently, logCPM values exhibit a mean-variance relationship that decreases based on count size, and the logCPM transformation roughly distorts the variance of RNA-seq counts as a function of count size, especially for genes with larger counts.

logCPM transformation can be analyzed using a trend approach for RNA-seq data analysis (149,150). However, this causes the mean-variance trend of low-count data to be ignored. Limma-trend models variance at the gene level, but RNA-seq count sizes can vary widely from sample to sample for the same gene. Due to different samples being sequenced at different depths, different count sizes can yield the same cpm values. Therefore, voom models the mean-variance trend of logCPM values at the individual observation level rather than applying gene-level variability to all samples within the same gene. To achieve this, the mean-variance trend of the logged read counts is estimated, and this mean-variance relationship is utilized to estimate the variance of each logCPM value. The estimated variance is then retained as an inverse weight for the logCPM value. The inverse square estimated standard deviation for each sample becomes the weight for that sample.

The voom method has accurately controlled the type I error rate and false discovery rate (69). The voom method produced results that were very close to the nominal type I error rate in scenarios considering equal or unequal library sizes. Moreover, voom consistently exhibited the lowest false discovery rate across various cut points. Notably, voom also showcased faster performance than alternative methods.

In the analysis of RNA-seq data, especially in DE analyses, normalization, and batch correction are applied to eliminate systematic biases and reduce variability. However, another factor complicating RNA-seq data analysis is the variability in sample quality. One strategy to model sample-specific variability involves excluding

high-variability samples from the dataset. This approach reduces variability but also reduces the power to identify DE genes. Alternatively, retaining all samples in the dataset allows for a comprehensive view but may limit the ability to distinguish true differences between experimental conditions from noise due to increased variation. To address this challenge, the concept of down-weighting observations from samples with high variability has been introduced. This approach aims to preserve maximum degrees of freedom while minimizing the impact of noisy observations (151). Consequently, sample weights, in addition to observational weights, can be determined post-voom transformation.

2.5.5. Feature Selection

Feature selection, which has several benefits, including removing redundant variables, decreasing time complexity, and enhancing the efficiency of many algorithms, is one of the most important challenges in high-dimensional data analysis. It is not recommended to make predictions using all features due to the possibility of overfitting. Given the sparsity assumption, it is important to choose the most important features, since most of the features do not influence the result. One of the most commonly used methods for feature selection is regularized regression methods, which shrinks regression coefficients to zero, leading to economic prediction models and dealing with the problem of overfitting (34).

Boruta algorithm (152) is a wrapper feature selection method derived from the Random Forest algorithm. This method tries to determine a threshold by taking advantage of the variable importance order used in the Random Forest algorithm. The set of variables is doubled with copy variables called ‘shadow variables’ from the copy of all variables. Random forest is trained on this new expanded dataset and variable importance values are created. A statistical test compares the significance of each real variable in the dataset with the maximum values of all dummy variables. Variables with significantly larger importance values are labeled important, respectively, while variables with smaller importance values are labeled unimportant. Thus, the Boruta algorithm checks at each iteration whether a real feature is of higher importance. All unimportant variables and shadow variables are removed. The previous steps are

repeated until all variables have been classified or a predetermined number of runs have been performed.

Variable selection with the Boruta algorithm has been used in gene expression (153) and microbiome (154) studies involving high-dimensional omics datasets. Boruta algorithm has been a recommended method for analyzing high-dimensional data as well as low-dimensional data (155).

2.6. Survival Modeling

The outcome variable is the survival time, which is the time until the event of interest occurs in many cancer studies. Various statistical methods can be used for analysis when the event of interest occurs in all individuals in the study. However, if the outcomes of the event become unobservable for individuals after a specific time point due to various reasons, or if individuals have not experienced the event by the end of the study, such instances are categorized as censored samples. This data type cannot be analyzed with standard statistical methods or machine learning-based prediction models developed for classification problems. This is due to the outcome variables in these data containing both event and time information (156). Therefore, survival algorithms have been developed to address this unique data type. Survival algorithms are concerned not only with whether the event of interest occurred but also when the event occurred.

2.6.1. Basic Concepts in Survival Analysis

Time-to-event variables: The survival data outcome variable comprises ‘status’ and ‘time’. The status variable represents the status of the individual at the end of the study, and the time indicates the duration of the follow-up period. The status variable is categorical, reflecting whether the individual experienced the event of interest. In cancer studies, this event is typically the time to death. However, events such as the time until cancer relapses, response to the treatment, disease development, or tumor disappearance can also be considered. The time until these events of interest are always continuous, positive, and usually exhibits a skewed distribution.

Censoring: Censored individuals are those who did not receive follow-up data during the study period. The true survival time of these uncensored patients is unknown.

Censored samples provide only partial details about the event's timing, leading to underestimating or overestimating real survival times (157). Censorship in survival studies may occur for a variety of reasons, including: i) individuals may still not have experienced the event of interest by the study's end, ii) follow-up may have been lost during the study period, iii) another untrackable event, such as death, may have occurred, iv) patients may have withdrawn from the study for various reasons. Three categories exist for censoring types: interval-, left-, and right-censoring (Figure 2.6). The observed survival time in data that has been right-censored is less than or equal to the true survival time, whereas the observed survival time in data that has been left-censored is higher than or equal to the true survival time. Data that has been interval-censored includes occurrences that take place inside a given time frame.

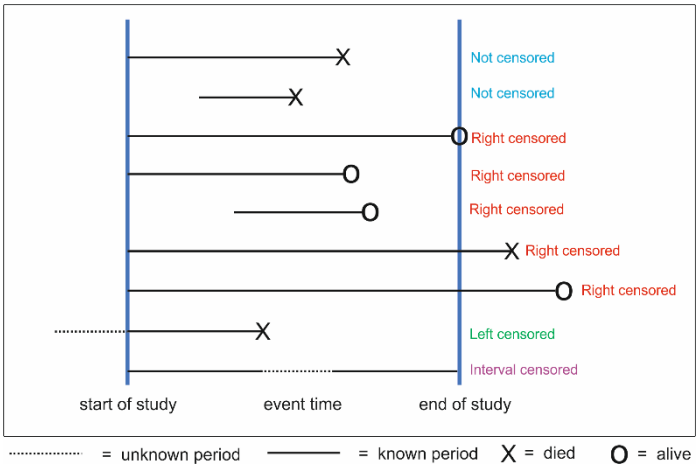


Figure 2.6. The three types of censoring.

Survival Data: Survival data, comprising n samples, can be described using a minimum of three variables for each sample, denoted as $X = \{(x_i, T_i, \delta_i)\}$, $i = 1,2,\dots,n$. Here, $x_i \in R$ represents the covariate vector of the i^{th} sample. T_i is the survival time if the i^{th} sample is uncensored, or T_i is the censoring time if the i^{th} sample is censored. T denotes the observed time until an event of interest occurs for uncensored data or the observed time to censorship for censored variables. T is non-negative and continuous. δ_i is the status variable of the i^{th} sample, taking a value of 1 for uncensored samples and 0 for censored samples. Using various functions, survival analysis predicts the time until the event of interest occurs for a new sample using covariate variables and estimates the survival probability at predicted survival time.

Survival Function (S(t)): The probability density function of T is denoted by $f_T(t)$. The cumulative distribution function of T, defined as $F_T(t)$, computes the probability that the event of interest (T) occurs before a specified time (t). The cumulative distribution function is given as follows.

$$F_T(t) = \int_{-\infty}^{t} f_T(u)du = P(T \leq t)$$

The survival function computes the probability that the time to the event of interest (T) is not earlier than a specified time (t). The survival function is given as follows.

$$S_T(t) = P(T \geq t) = 1 - F_T(t) = \int_{t}^{\infty} f_T(u)du$$

The relationship among $f_T(t)$, $F_T(t)$ and $S_T(t)$ are shown in Figure 2.7. Due to $lim_{t\to\infty}F_T(t) = \infty$, $S_T(\infty) = 0$.

The survival function is non-increasing and monotonically decreases with t (Figure 2.8). Since all samples survive at the beginning of the study, having not experienced the event of interest, the initial value of the survival function at the origin is 1 when t=0. ($S_T(0) = 1$).

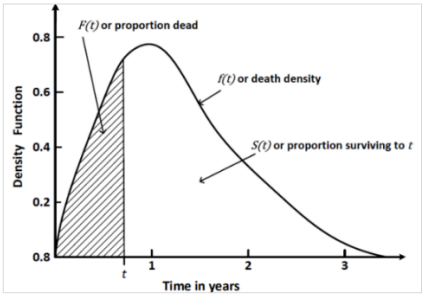


Figure 2.7. The relationship among functions, which are f(t), F(t), S(t).

Probability Density Function (F(t)), or the Cumulative Incidence Function (R(t)): The probability that an individual has a survival time equal to or less than t time.

Hazard Function (h(t)): The hazard function does not calculate a probability. This function is the rate of the event at a specified time (t). The hazard function determines the instantaneous failure rate at time t, provided an individual has survived until t, and is defined by

$$h_T(t) = \lim_{\Delta t \to 0} \frac{P(t \le T < t + \Delta t | T \ge t)}{\Delta t} = \frac{f_T(t)}{1 - F_T(t)}$$

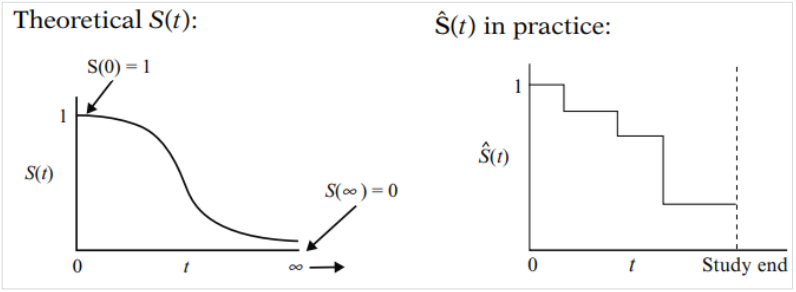


Figure 2.8. Smooth curve and stepped line graphs for survival function.

h_T is non-negative and has no upper bound. If no event happened in δt, then $h_T(t) = 0$. The hazard function may exhibit various graphical shapes, as shown in Figure 2.9.

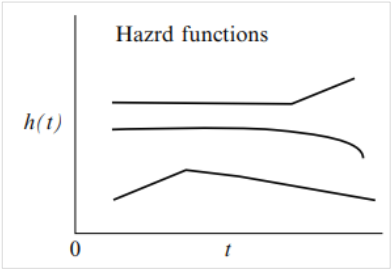


Figure 2.9. Hazard functions.

Cumulative Hazard Function (H(t)): The cumulative hazard function is the total amount of probability accumulated up to time *t*, where 'instantaneous probability' is derived from the probability distribution function. It is the integral of the hazard function from time 0 to time *t* and is also equal to the AUC of the *h(t)* from time 0 to time *t*.

$$H(t) = \int_0^t h_T(u)du$$

2.6.2. Statistical Methods for Survival Analysis

When analyzing survival data, three fundamental approaches are employed: non-parametric, semi-parametric, and parametric, depending on the research question (Figure 2.12). A comparison of these approaches is given in Table 2.3.

Parametric Approaches

Parametric survival models assume that the survival times or the logarithm of the survival times for all individuals in the data follow a theoretical survival distribution. These models produce survival estimates based on this distribution (158). The most common parameter estimation method in these models is the maximum likelihood estimation method. Parametric approaches offer an advantage in reliably estimating survival times, especially for events occurring long after the observed data. Among the various parametric distributions, each employes different hazard functions; the most commonly used ones include (i) Exponential, (ii) Weibull, (iii) Gompertz, and (iv) Log-logistic.

Table 2.3. Comparison of type of survival approaches.

Type of Approaches	Advantages	Disadvantages
Non-parametric	-It is used when the theoretical distribution of survival times is unknown or the proportional hazard assumption does not hold. -It's flexible.	-Less effective results if survival times are theoretically distributed. -Survival function has piecewise constants instead of being smooth. It can give unrealistic estimates with small sample sizes.
Parametric	-It is easy to interpret as survival times show a theoretical distribution. -It's simple, efficient, and effective.	-It may give inaccurate results when distribution assumptions are not met.
Semi-parametric	- It does not need distribution information for survival times.	-Outcome variable is difficult to interpret as its distribution is unknown.

Exponential Distribution: The Exponential distribution is the simplest parametric model, characterized by a single parameter, λ, where the mean of this distribution is also λ. It assumes that the random events of failure and death are time-independent, with a constant instantaneous hazard over time. The probability density function is given by $f(t) = \lambda \exp[-\lambda t]$, the instantaneous hazard function is $h(t) = \lambda$, the cumulative hazard function is $H(t) = \lambda t$, and the survival function is $S(t) = \exp[-\lambda t]$ (Figure 2.10).

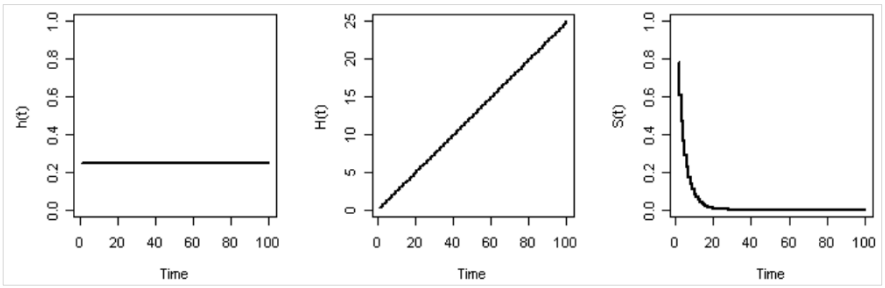


Figure 2.10. Exponential distribution, when λ=0.25.

Weibull Distribution: The Weibull distribution is characterized by two parameters: a scale parameter, λ, and a shape parameter, γ. The probability density function is given by $f(t) = \lambda\gamma t^{\gamma-1} \exp[-\lambda t^{\gamma}]$, the instantaneous hazard function is $h(t) = \lambda\gamma t^{\gamma-1}$, the cumulative hazard function is $H(t) = \lambda t^{\gamma}$, and the survival function is $S(t) = \exp[-(\lambda t)^{\gamma}]$. The behavior of the instantaneous hazard concerning time depends on the value of γ; it monotonically decreases over time when γ <1, remains constant when γ =1, and increases over time when γ >1 (Figure 2.11).

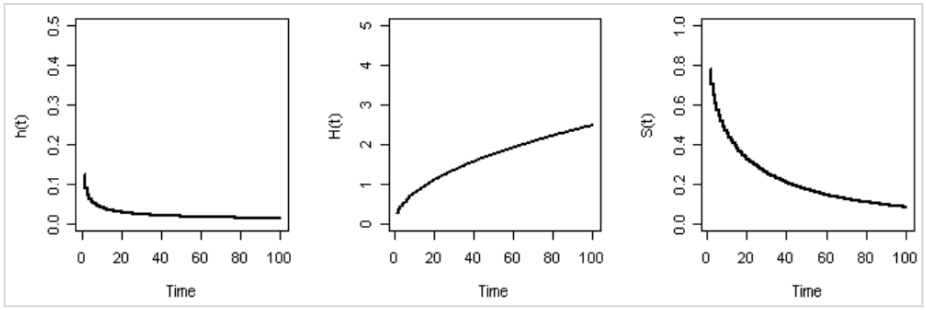


Figure 2.11. Weibull distribution for λ=0.25 and γ =0.5.

Gompertz Distribution: The probability density function is given by $f(t) = \lambda \exp[\gamma t] \exp[-\lambda/\gamma\,(\exp[\gamma t] - 1)]$, the instantaneous hazard function is $h(t) = \lambda \exp[\gamma t]$, the cumulative hazard function is $H(t) = \lambda/\gamma\,(\exp[\gamma t] - 1)$, and the survival function is $S(t) = \exp[-\lambda/\gamma\,(\exp[\gamma t] - 1)]$.

Logistic Distribution: For the logistic distribution, the hazard function behaves non-monotonically. The survival time is denoted by *T*, μ is the parameter that determines the location of the function, and σ is the scale parameter. The probability density function is given by $\frac{e^{-(t-\mu)/\sigma}}{\sigma(1+e^{-(t-\mu)/\sigma})^2}$, the survival function is $\frac{e^{-(t-\mu)/\sigma}}{1+e^{-(t-\mu)/\sigma}}$, and the instantaneous hazard function is $\frac{1}{\sigma(1+e^{-(t-\mu)/\sigma})}$.

Log-logistic Distribution: Similar to logistic distribution, the hazard function behaves non-monotonically for the log-logistic distribution. The survival time is log(*T),* and γ>0 is the shape parameter. The probability density function is given by $f(t) = \lambda\gamma t^{\gamma-1}/(1+\lambda t^{\gamma})^2$, the instantaneous hazard function is $h(t) = \frac{(\lambda\gamma t^{b-1})}{1+\lambda t^{\gamma}}$, the cumulative hazard function is $H(t) = \log(1+\lambda t^{\gamma})$, and the survival function is $S(t) = (1+\lambda t^{\gamma})^{-1}$.

a. Linear Regression Models

Tobit regression employs a linear regression with a Gaussian distribution (159). The Buckley and James regression utilizes a least-squared estimator for censored dependent variables (160). In a particular study, this method was combined with the elastic net regularizer (161). Penalized regression selects variables and estimates the coefficient simultaneously (162). It addresses challenges related to multicollinearity and high dimensionality. Various types of penalized regression include weighted regression (163) and structured regularization (164).

b. Accelerated Failure Time (AFT) Model

The accelerated failure time model has some assumptions (165). It assumes the linear relationship between the logarithm of the survival time and the covariates. Additionally, it assumes that the features have a multiplicative effect on the survival time.

Non-parametric Approaches

Non-parametric methods offer an alternative to parametric approaches by avoiding assumptions about the distribution of event times. These methods typically produce descriptive statistics, laying the groundwork for subsequent parametric or semi-parametric analyses. Non-parametric techniques are particularly valuable when no suitable theoretical distribution adequately fits the data.

a. Kaplan-Meier (or Product-Limit) Estimator

The non-parametric, the Kaplan-Meier estimator, is employed to estimate the survival distribution function from survival data (20). Kaplan-Meier divides time into

intervals, determined by observed event time rather than predefined intervals. Individuals who have not yet experienced the event at the interval's start and have not been censored during the interval or earlier ones are considered at risk for each interval. These at-risk individuals are estimated to have a survival probability and contribute to the prediction of survival probability until the event occurs or they are subject to censorship. The number of survivors is divided by the number of at-risk patients to calculate the survival probability. The cumulative probability of survival up to time interval *t* is then calculated by multiplying the survival probabilities across all preceding time intervals.

Let $t_j, j = 1,2, \dots, n$ represent the total set of failure times recorded, and T be the maximum failure time. The Kaplan-Meier estimator of the survival function, denoted as $S(t) = P(T \geq t)$, is expressed as follows.

$$\hat{S}(t) = \prod_{j:t_j \leq t} \left(1 - \frac{d_j}{r_j}\right), \quad 0 \leq t \leq T$$

where d_j is the number of individuals who experienced the event at the time t_j, and r_j is the number of individuals in the risk set just before the time t_j.

Kaplan-Meier curves represent the Kaplan-Meier estimator of survival probability over time. These curves start from 1 and decrease over time as a stepped line instead of a smooth curve. This is because cumulative survival decreases at the precise time a death occurs and remains flat between successive death times (Figure 2.8).

The log-rank test, also known as the Mantel log-rank, the Cox Mantel log-rank, or the Mantel-Haenszel test, is widely used for comparing the Kaplan-Meier curves of two or more samples. This test assesses whether the survival distributions of different samples are equal. The underlying assumption is that the hazard functions of the samples are parallel. It is a large-sample chi-square test, which calculates observed versus expected cell counts over categories of outcomes. The log-rank test takes each time point with a failure event, creating 2x2 tables that display the number of individuals who experienced the event of interest and the total number of individuals under follow-up. For each table, calculations are performed for observed deaths, expected deaths, and the variance of the predicted number. These values are then

summed across all tables, yielding a chi-square statistic with 1 degree of freedom. The null hypothesis for the log-rank test posits that "The samples have identical distribution curves.", while the alternative hypothesis suggest that "The samples have different distribution curves." Alternative tests like Wilcoxon (Breslow), Tarone-Ware, Peto, and Flemington-Harrington can be substitutes for the log-rank test.

b. Life-Table (Actuarial or Cutler-Ederer) Estimator

The Life-Table estimator approximates the Kaplan-Meier estimator, particularly in large-scale population surveys (23). This method assumes that the failure rate within a given interval remains consistent across all subjects and is independent of the probability of survival in other time periods.

c. Nelson-Aalen Estimator

The Nelson-Aalen estimator is based on the counting process approach and predicts the cumulative hazard function (22). The cumulative hazard at time *t* is below.

$$\widehat{H}(t) = \prod_{j:t_j \leq t} \frac{d_j}{r_j}, \;\; 0 \leq t \leq T$$

Various equations can be used when converting to a survival function, such as $H(t) = -log[S(t)], S(t) = e^{-H(t)}$.

Semi-Parametric Approaches

Semi-parametric approaches are based on regression analysis approach. Therefore, some assumptions exist, like the proportional hazards. Parameter estimation is performed using partial likelihood. The reason why these approaches are called semi-parametric is that the distribution of the outcome is not known.

a. Cox Proportional Hazard

The most common model used to analyze survival data is the Cox proportional hazards model (157). In this model, all individuals have the same proportion of hazards

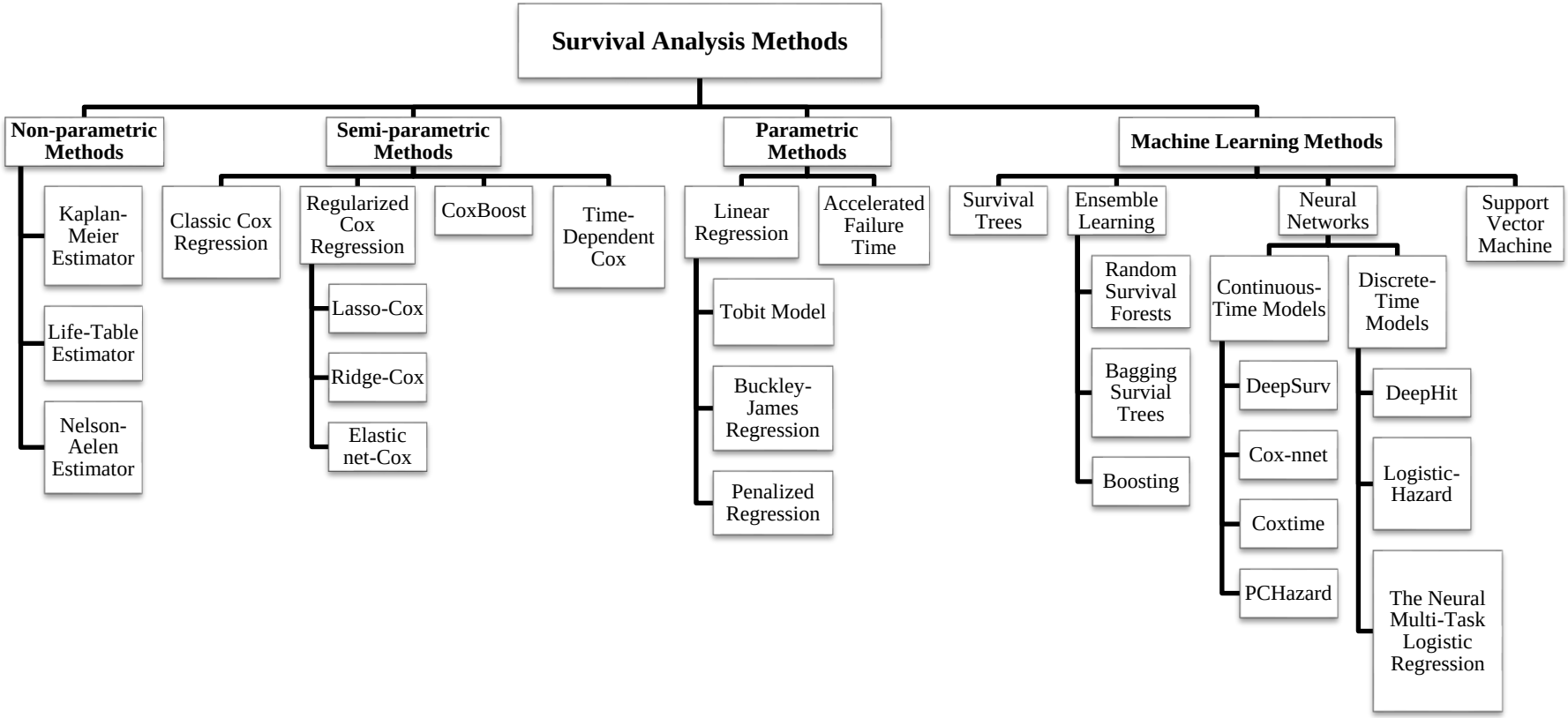


Figure 2.12. Survival analysis methods.

at all times and the hazard ratio is maintained over. The unspecified baseline hazard makes this model semi-parametric. The model is based on hazard function, denoted as $h(t|x)$, which is the probability that an individual with predictors *x* will experience an event at time *t*, given that the individual is alive just before *t*.

The Cox proportional hazards model relies on several assumptions (157). First, there's the proportional hazards assumption, which states that the hazard ratio won't change during the course of the follow-up. As an example, a Cox proportional hazard model that uses the patient's sex as the predictor variable makes the assumption that the risk is the same for males and females over the course of the follow-up. The second assumption is the independence of survival times. According to this assumption, the survival time of one patient does not depend upon the survival time of another. Thirdly, a linear relationship between time-independent covariates and the log hazard should exist. Lastly, censoring is assumed to be uninformative about the outcome of interest. Those who are censored are exposed to the same risk at the end point of the study as those who continue to be monitored (166).

Cox Proportional Hazards model is described as follows:

$$h(t) = h_0(t)e^{(\beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p)}$$

$h(t)$ is the expected hazard at time *t*. $h_0(t)$, is the baseline hazard function. $X_p = (X_1, X_2, \dots, X_p)$ is the covariates. $\beta^T = (\beta_1, \beta_2, \dots, \beta_p)$ is the coefficients.

Cox model models partial likelihood using maximum likelihood:

$$pl(\beta) = \prod_{i=1}^{n} \left(\frac{exp^{\beta X_i}}{\sum_{j \in R_i} exp^{\beta X_i}} \right) \quad (1)$$

Although the Cox proportional hazards model is extensively used in survival analysis, it has some disadvantages. Firstly, the assumption that hazards are proportional over time cannot always hold. Secondly, the model follows a restrictive parametric format concerning how variables influence the outcome (167). It also has limitations, particularly when dealing with high-dimensional data, where model assumptions are frequently broken. Interpreting results becomes challenging, particularly in interactions (168). Given these limitations, alternative survival approaches have been developed, especially for high-dimensional genetic data.

2.7. Survival Modeling of High-Dimensional Data

Cox models are linear models that make assumptions about the hazard ratio, and these assumptions may be violated in real-world data. That is, the Cox model can only model linear interactions. Semi-parametric methods and parametric methods, rely on the likelihood or partial likelihood functions commonly used in clinical studies. However, there is a growing need for methods that perform better in more complicated data, which may include high dimensional and non-linear relationships, for example, in RNA-seq data with a number of features that exceed the number of samples.

Machine learning algorithms commonly used for survival analysis are extensions of those developed for classification problems or traditional survival models (Figure 2.12). Several machine learning methods have been adapted to address survival analysis problems, offering enhanced prediction performance. These approaches can capture complex and nonlinear relationships, and unlike the Cox proportional hazards (PH) model or penalized methods, they do not strictly require the Cox PH assumption. As a result, machine learning methods can provide more accurate survival predictions, especially for high-dimensional and complex datasets.

2.7.1. Penalized Likelihood Cox Models

In classical data, when the number of variables (p) is less than the number of samples (n), linear regression models perform well; however, when the number of variables equals or exceeds the number of samples, they perform poorly. In these cases, it becomes challenging to model with all features, and overfitting may result in poor results (169). Gene expression data, which measure the expression levels of millions of genes, falls into the high-dimensional data category. Therefore, the standard Cox partial likelihood method cannot be applied directly to obtain parameter estimation in such data with abundant variables. In addition to the high dimensionality, the expression levels of some genes are often highly correlated, leading to the problem of high collinearity (Figure 2.4). To address these challenges, penalized regression models was developed by applying various penalties to the linear regression model. The term ‘penalized’ implies adding constraints to the model due to its numerous variables. Through penalization, coefficient values are shrunk, and some may be reduced to 0. Lambda (λ), a tuning parameter, determines the extent of shrinkage. This

process ensures that less-contributing variables have coefficients close to or equal to 0, revealing the most relevant features for the outcome variable. Penalized methods are also known as shrinkage or regularization methods and serve as feature selection methods. Various regularized Cox models have been developed.

Lasso-Cox

Lasso (Least Absolute Shrinkage and Selection Operator) regression is a linear model incorporating a regularization term into the loss function, emphasizing sparse coefficient prediction. It predicts regression coefficients through shrinkage and performs feature selection simultaneously. The penalty term in Lasso is called the L1-norm, which denotes the sum of the absolute coefficients. By minimizing this penalty, Lasso can yield coefficients that are precisely 0, provided that the sum of the absolute values of the coefficients is below a certain constant. Consequently, the model's complexity is diminished, making it a viable alternative to subset selection methods for variable selection (170).

The L1-norm penalty term in Lasso regression has been integrated with log-partial likelihood, making it applicable as a survival algorithm (34). Several studies have also existed using Lasso with gene expression data for survival analysis (34,171,172).

Ridge-Cox

Like Lasso regression, Ridge regression aims to shrink the regression coefficients, bringing the coefficients of variables with minimal contribution to the outcome close to 0. However, unlike Lasso, which employs the absolute value of coefficients in its penalty term, Ridge uses the square of the coefficients. Ridge's penalty term is called the L2-norm, representing the sum of the squared coefficients. The magnitude of the penalty, denoted by a constant λ, determines the extent of shrinkage. When $\lambda=0$, the penalty has no effect, and ridge regression produces classical least squares coefficients. As λ increases, the impact of the shrinkage penalty gets larger, causing the ridge regression coefficients to approach 0. Ridge regression shrinks the coefficients towards zero without precisely setting any of them to zero (35).

The L2-norm penalty term in Ridge regression was integrated with log-partial likelihood and began to be employed as a survival algorithm (169).

Elastic net-Cox

Elastic-net (EN) is a method that combines L1-norm and L2-norms by penalizing tuning parameters. It performs the feature selection while simultaneously addressing the correlation among features (36). The EN penalty term in the log-partial likelihood function has been employed to analyze survival data (25).

2.7.2. CoxBoost

CoxBoost is an offset-based boosting approach (173). This approach predicts Cox proportional hazard models through flexible penalization of covariates, allowing unrestricted estimation of essential covariate parameters.

2.7.3. Survival Trees

Decision trees are a non-parametric supervised learning algorithm for regression or classification problems (174). Their input and output variables can be both categorical and continuous. Decision trees effectively partition complex and heterogeneous datasets into homogeneous subgroups (nodes in the tree), utilizing simple predefined decision rules based on a specified target variable. The outcome is a hierarchical structure of candidate nodes extending from the tree's root to terminal nodes, also known as leaves. The root is the initial node at the top of the tree, encompassing all samples. Subsequent nodes or internal nodes branch off from the root, forming a tree structure with each node contributing to the classification of samples. The more nodes, the more complex the model becomes. There are leaf nodes or leaves at the end of the decision tree that give the final output. Tree-based methods can vary regarding splitting rule, pruning mechanism, ensembles, and randomization.

Various decision tree algorithms are available, including CART, ID3, and C4.5. ID3 and C4.5 are particularly effective tools for both classification and regression tasks. The CART (Classification and Regression Tree) algorithm is one of the first algorithms developed (174). A CART tree algorithm starts with the root node containing all samples, makes a comprehensive search through all potential binary

splits based on covariates, and selects the best one according to a splitting rule based on an appropriate measure. It recursively partitions the training dataset into smaller subsets, predicting a categorical or continuous outcome variable (*Y*) based on covariates $X = (X_1, \ldots, X_p)$. The splitting rule is based on maximizing intra-node homogeneity or inter-node heterogeneity. For an X covariate, a split has the form $X<c$ and its indicator function $I(F<c)$ is defined for each sample, where c is a split threshold value to divide all samples into two subsets. These two subsets created are the daughter nodes of the current node. The best splitting number is maximized by specific splitting rules. The result is a disjoint subset (end node = terminal node). The predictions are uniquely assigned to the end node that a test sample belongs to. However, in situations where noise exceeds true signals or unmeasured factors are present, there is a risk that the single tree method may incorrectly split terminal nodes, leading to a large and complex tree (175). The algorithm chooses a feature and a threshold at each tree node to divide the data in half. Until the sample size of one node is small enough, this procedure is repeatedly applied to the two daughter nodes and next nodes. The best feature and threshold are chosen using metrics such as the Gini index to provide the best possible discrimination. Unlike other tree-based methods, the CART algorithm consistently generates a binary tree.

Another important component of the CART algorithm is the stopping criterion. A good selection of the stopping criterion ensures that the final tree is good. Excessively small or large trees may fail to generalize to test data, resulting in underfitting or overfitting issues in the training dataset. To mitigate overfitting, reduce the tree size, and minimize prediction errors in tree-based algorithms, a pruning and selection method is employed either during or after the tree creation process. This involves removing partitions that do not significantly contribute to classification (174). As a result of pruning, selecting a single tree from the subtree array is necessary. Various methods, including cross-validation, bootstrap, AIC/BIC, and graphical ("kink" in the curve or elbow), can be employed for selection (176). Numerous pruning methods in the literature, including cost complexity pruning, critical value pruning, pessimistic pruning, Minimum Description Length (MDL) pruning, and many others (174,175,177). The tree continues to split into two at each node until the stopping criteria are met.

For survival analysis, survival trees are an extension of decision trees. Regression analysis and predictions based on censored survival are made possible by them. Survival trees contract a decision tree by iteratively partitioning it into tree nodes based on specific features. Like standard trees, each division utilizes a dissimilarity measure that computes the disparity in survival between two new nodes and chooses the best partition that maximizes this difference. Various dissimilarity measures, including log-rank test statistics, are employed for survival analysis. Different approaches to splitting and pruning have been used in methods utilizing tree structures for survival data, as outlined in Table 2.4.

Assume that *U* represents the true survival time, and *C* is the censoring time for applying tree-based algorithms to survival data. The variable $\tau = \min(U, C)$ represents the time until the event occurs or the individual is censored. The variable $\delta = I(U \leq C)$ takes the value 1 if the true time-to-event is observed and 0 if the individual is censored. $X = (X_1, \ldots, X_p)$ denotes the vector of covariants. The initial concept of applying tree-based algorithms to censored data was introduced by Ciampi et al. (178) and Marubini et al. (179) but was further developed by Gordon & Olshen (41).

Randomness can badly affect tree-based methods as the tree grows with randomly selected individuals through bootstrapping. Developing a single tree may yield different prediction results. Ensemble methods, on the other hand, treat each tree independently, employing a random set of explanatory variables at each node and ultimately considering all the results. The basic idea is that combining multiple survival tree estimators yields better predictions than a single independent tree. This enhances the predictive performance compared to individual decision trees. Growing a full-size tree for each bootstrap sample also mitigates issues related to pruning and selection. Averaging the results of multiple trees helps reduce overfitting (180).

2.7.4. Bagging Survival Trees

The high variance problem may arise in decision trees since different randomly selected train samples are used, and quite different estimates are obtained. Also,

Table 2.4. Splitting and pruning rules in survival trees.

Author(s)	Splitting rule	Pruning rule
Gordon and Olshen (1985) (41)	Impurity reduction, using the Wasserstein distance between Kaplan-Meier survival curves	Cost-complexity pruning and cross-validation
Ciampi, Thiffault, Nakache, and Asselain (1986) (181)	Two-sample test statistics based on the weights such as log-rank test statistic	Akaike information criterion (AIC)
Segal (1988) (182)	Two-sample test statistics based on the weights such as log-rank test statistic	Not available
Butler, Gilpin, Gordon, and Olshen (1989) (183)	Two-sample test statistics based on the weights such as log-rank test statistic	A within-node measure
Davis and Anderson (1989) (184)	Exponential log-likelihood loss	Cost-complexity pruning
Therneau, Grambsch, and Fleming (185)	The martingale residuals from a null Cox model	Cost-complexity pruning and crossvalidation
LeBlanc and Crowley (1992) (186)	The node deviance measure for the proportional hazards model calculating the full likelihood by the Nelson-Aalen estimator	Cost-complexity pruning and crossvalidation
Keles and Segal (2002) (187)	A survival tree based on the square error of the martingale residuals from a null Cox model	
LeBlanc and Crowley (1993) (188)	Two-sample test statistics based on the weights such as log-rank test statistic	Resampling and permutation
Intrator and Kooperberg (1995) (189)	Two-sample test statistics based on the weights such as log-rank test statistic	Cost-complexity pruning
Zhang and Singer (1999) (190)	A combination of impurity of the censored samples and impurity of the observed time	Cost-complexity pruning
Breiman (2002) (191)	Probability .75 to split on time, and Probability .25 to split on a covariate	N/A (embedded within the survival forest algorithm)
Molinaro, Dudoit, and van der Laan (192)	An inverse probability of censoring weighted (IPCW) loss function	Cost-complexity pruning and crossvalidation
Jin et al. (2004) (193)	A splitting rule based on the variance of survival times	
Hothorn et al. (2006) (194)	Minimum p value	Stop when no p value is below a prespecified a-level

allowing decision trees to grow to maximum depth can cause an overfitting problem. Bagging improves prediction accuracy and reduces the prediction variance using the bootstrap algorithm, which takes the mean from multiple bootstrap samples from the training datasets and fits the decision tree to each samples. Breiman (195) created the bagging procedure to solve the overfitting and stability problems encountered in single decision trees.

Bagging survival trees are calculated from survival trees based on bootstrap samples. For this, a survival tree is built on each bootstrap sample. For each subsample, the bootstrap aggregated estimator of the survival function is the Kaplan-Meier curve. Finally, the mean of predictions from those bootstrap samples is calculated. In survival analysis, the bagging procedure was applied to the right censored data by Hothorn et al. (45). They used bagging with decision trees and predicted ensembling outputs via the Kaplan-Meier curve for lymphoma and breast cancer patients.

The disadvantage of the bagging procedure is that it requires more time and resources to create more than one training set. Also, Bagging can improve the accuracy of the model by reducing variance, but it cannot solve the problem of highly correlated trees.

2.7.5. Random Survival Forests

Since trees are created based on the same set of predictions in the bagging algorithm, strong predictors are likely to be selected repeatedly. Accordingly, averaging these predictions may not reduce the variance much, as bagging can generate similar trees that produce highly correlated predictions. Like the bagging algorithm, the random forest algorithm produces multiple trees but also considers the correlation of predictions from those samples (38). The random forests algorithm takes *m* of estimators to be evaluated in internal nodes and chooses the best instead of considering all estimators each time. The number *m* is usually the square root of the features. Thus, the correlation between trees decreases, and hence the variance decreases.

The difference between the random forests algorithm and the bagging procedure is that it chooses a random sample among the predictive variables. The prediction from the random forests algorithm is obtained by averaging hundreds or thousands of trees that differ from each other. Because random forests average many trees, they can reduce overfitting over single-decision trees. Thus, it creates a more robust and sophisticated mode than a single tree. These algorithms can also capture nonlinear effects and interaction terms. It can also deal with multiple interrelated variable states in data with collinearity problems.

The random survival forests algorithm was created by modifying the random forest algorithm for survival data. The random forests algorithm has been adapted to the survival responses by Breiman (191), Hothorn et al. (46), and Ishwaran et al. (39). This algorithm applies two-step randomization to increase the prediction performance according to a single decision tree. First, a bootstrap random sample is taken for the growth of each tree. Second, each tree node randomly selects some explanatory variables (39). Several independent bootstrap samples are drawn randomly from the training set. These samples are the same size and obtained by the substitution method. Each bootstrap sample contains an average of approximately two-thirds of the dataset. The remaining one-third is called out-of-bag data, which will not appear in the bootstrap sample. A separate decision tree grows according to a particular splitting rule without pruning from each bootstrap sample. Using bootstrap data prevents overfitting. The second randomization is done at the node separation level. At each tree node, the *p* variable is randomly selected. Each node is separated using one of the variables that maximizes the difference in survival between daughter nodes. Each tree grows under the constraint of a terminal node until a specific stopping rule is met.

For each tree, the cumulative hazard function is calculated with an estimator such as Kaplan-Meier or Nelson-Aalen. All samples in the same node have the same cumulative hazard function. The mean of each tree's calculated cumulative hazard functions in the forest forms the ensemble cumulative hazard function. The algorithm then calculates the estimation error of the cumulative hazard estimation for the out-of-bag data (196).

However, random forests have the disadvantage of being unable to interpret a single tree because they average various trees. Also, this algorithm has computational and cost problems as too many trees are formed.

2.7.6. Boosting

The bagging and random forests algorithms use independent trees, while the boosting algorithm builds trees based on previous trees. That is, the residuals at each state are used to grow sequential trees. The boosting algorithm iteratively combines weak learners to create a strong learner that can predict more accurate outcomes. AdaBoost (Adaptative Boosting Algorithm) is one of the most popular boosting

applications (197). This method works iteratively, identifying misclassified data points and adjusting their weights to minimize training errors. The model iteratively optimizes until it produces the most robust predictor. Apart from that, there are XGBoost (198), GardientBoost (199), and BrownBoost (200). Gradient boosting works by sequentially adding estimators to a collection, each correcting the errors of the previous one. Yet, gradient boosting makes use of the residual errors of the prior predictor rather than altering the weights of the data points as AdaBoost does. Because it combines the boost method with the gradient descent algorithm, it is referred to as gradient boost. XGBoost (Extreme gradient boost) is a gradient boosting app designed for computation speed and scale. XGBoost takes advantage of multiple cores on the CPU, allowing learning to occur in parallel during training.

The boosting algorithm was applied to censored data, which iteratively combines base learners to obtain strong learners (194).

Since it is necessary to tune the learning rate, the tree depth, and the minimum number of observations in terminal nodes in addition to the number of repetitions in the boosting algorithm, having too many hyperparameters is a disadvantage.

2.7.7. Survival Support Vector Machine

The Support Vector Machine (SVM) is used in classification and regression problems (201). SVM works very well with high-dimensional data by avoiding the curse of dimensionality problems. The SVM algorithm finds a hyperplane in an *N*-dimensional space, and this hyperplane classifies the data points. There are many possible hyperplanes to separate the two classes of data points; however, the main objective is to find a plane with a maximum distance between the data points of both classes. This maximum distance is called a margin. This margin is calculated using data points known as support vectors.

A hyperplane equation is given below

$$y = w^T x + b$$

In this equation, output *y* indicates whether it is in a positive or negative class. *w* represents the coefficients, and b is the constant value. The SVM algorithm is an

optimization problem; a loss function must be minimized. The problem is formulated as follows

$$\min_{w,b,\varepsilon,\varepsilon*} \frac{1}{2} w^T w + \gamma \sum_{i=1}^{n} (\varepsilon_i + \varepsilon_i^*),$$

$$subject\ to \begin{cases} w^T \varphi(x_i) + b \geq y_i - \varepsilon_i, & \forall i = 1, \dots, n \\ -(w^T \varphi(x_i) + b) \geq -y_i - \varepsilon_i^*, & \forall i = 1, \dots, n \\ \varepsilon_i \geq 0, & \forall i = 1, \dots, n \\ \varepsilon_i^* \geq 0, & \forall i = 1, \dots, n \end{cases}$$

For a new x^* point where α_i and α_i^* are Lagrange multipliers, the index is found by the formula

$$\hat{y}(x^*) = \sum_i (\alpha_i - \alpha_i^*) \varphi(x_i)^T \varphi(x^*) + b$$

If the data has a higher dimensional feature space, a kernel function is used to find a classifier to separate the two classes. The main advantage of SVM is that it can consider the complex, non-linear relationships between features and survival with the kernel trick. A Kernel function is shown as $k(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$. $k(x, y) = x^T z$ is used for the linear kernel, $k(x, z) = (\tau + x^T z)^a, \tau \geq 0$ is used for the polynomial kernel of degree a, $\mathrm{k(x,z)} = \exp(-\frac{||\mathrm{x-z}||_2^2}{\sigma^2})$ is used for the RBF kernel.

As a result of its successful results in regression and classification problems, the SVM algorithm has also been extended for survival data. Different approaches have been adopted to use the standard SVM algorithm in survival analyses. Shivaswamy et al. (202) adopted the support vector regression approach, while Van Belle et al. (203) and Evers & Messow (204) applied SVM based on ranking constraints. Since outcomes were uncertain for censored data, all censored samples were removed in the earliest support vector regression approaches, or censored samples were considered non-events. These situations caused either underestimated failure times or biased models. However, the support vector regression model proposed by Shivaswamy et al. was formulated as follows

$$\min_{w,b,\varepsilon,\varepsilon*} \frac{1}{2} w^T w + \gamma \sum_{i=1}^{n} (\varepsilon_i + \varepsilon_i^*),$$

$$subject\ to \begin{cases} w^T \varphi(x_i) + b \geq y_i - \varepsilon_i, & \forall i = 1, \dots, n \\ -\delta_i (w^T \varphi(x_i) + b) \geq -\delta_i y_i - \varepsilon_i^*, & \forall i = 1, \dots, n \\ \varepsilon_i \geq 0, & \forall i = 1, \dots, n \\ \varepsilon_i^* \geq 0, & \forall i = 1, \dots, n \end{cases}$$

For a new x^* point where α_i and α_i^* are Lagrange multipliers, the index is found by the formula

$$u(x^*) = \sum_i (\alpha_i - \delta_i \alpha_i^*) \varphi(x_i)^T \varphi(x^*) + b$$

Van Belle et al. (203) and Evers & Messow (204) considered and formulated the survival data as a ranking problem. In this approach, instead of dealing with the prediction of the survival time, it is concerned with whether the patient's risk of the event is high or low so that appropriate treatment can be given. The method includes a penalty for each pair of comparable data points where the order in the prognostic index differs from the observed order. The comparison indicator for $\{(x_i, y_i, \delta_i), (x_j, y_j, \delta_j)\}$ sample pairs is as follows

$$comp(i,j) = \begin{cases} 1 & if\ \delta_i = 1\ and\ \delta_j = 1 \quad \delta_i = 1\ and\ \delta_j = 0\ and\ y_i \leq y_j \\ 0 & otherwise \end{cases}$$

The model is formulated as follows

$$\min_{w,\varepsilon} \frac{1}{2} w^T w + \gamma \sum_{i=1}^{n} \sum_{\substack{j:\, y_i > y_j \\ comp(i,j)=1}} \varepsilon_{ij},$$

$$subject\ to \begin{cases} w^T (\varphi(x_i) - \varphi(x_j) \geq 1 - \varepsilon_{ij}, & \forall i = 1, \dots, n;\ \forall j :\ y_i > y_j\ and\ comp(i,j) = 1 \\ \varepsilon_{ij} \geq 0, & \forall i = 1, \dots, n;\ \forall j :\ y_i > y_j\ and\ comp(i,j) = 1 \end{cases}$$

For a new x^* point where α_{ij} is Lagrange multipliers, the index is found by the formula

$$u(x^*) = \sum_{i=1}^{n} \sum_{\substack{j:\, y_i > y_j \\ comp(i,j)=1}} \alpha_{ij} (\varphi(x_i) - \varphi(x_j))^T \varphi(x^*)$$

2.8. Stacking Idea

The stacking idea presents mechanisms that can be considered classification and regression problems for survival problems (61). This idea converts survival data with time and status variables into classification data with binary outcome variables. Thus, all regression and classification algorithms can be applied to the new dataset because it doesn't include time and status variables.

There are some advantages of transforming survival data with stacking and making it analyzeable with classification algorithms. Firstly, the number of algorithms such as boosting, random forests, and deep neural networks developed primarily for classification problems is considerably more than those developed for survival analysis. Thus, numerous high-performance classification algorithms that cannot be directly applied to survival data are also made available for survival analysis. Secondly, the algorithms created for classification can also be made available for survival data with an additional study due to differences in survival and classification data structures during survival analysis. For example, a random survival forest algorithm was again adapted to survival data using the random forest algorithm so that the random forest algorithm used in classification problems can be used in survival problems (205). The stacking idea will be important in adapting existing and future classification algorithms to survival analysis. Thirdly, the transformation with the stacking idea can be an advantage in providing higher performance than the standard linear Cox model in some cases, especially in complex effects such as interactions in survival data (168). Finally, the Cox proportional hazards standard survival model is a linear model that assumes the relationship between covariates and hazard is constant over time, and this assumption is not always possible.

To consider the survival problem as a classification problem, the sequential in time structure of partial likelihood is used. In the standard Cox proportional hazards model, the β coefficients are chosen by maximizing the partial likelihood. The reason for using the term “partial” likelihood is that only the likelihood of individuals

experiencing the relevant event is considered in the probability formula, and the likelihood of censored individuals is not fully considered. That is, the Cox model does not consider all individuals' likelihood. The partial likelihood can be written as a product of probabilities.

$$L = L_1 * L_2 * L_3 * \dots * L_k = \prod_{j=1}^{k} L_j$$

k is the number of failure times. L_f gives the likelihood of failure at the f^{th} failure time. At the f^{th} failure time, the set of individuals at risk is the risk set and is denoted by $R(t_{(f)})$. Partial likelihood focuses on individuals who experienced the event of interest. Also, it considers the survival time until censored for censored individuals. That is, during the calculation of L_f, the contribution of this censored individual, who was censored after the f^{th} time of failure, is also included in this calculation (157). Since the partial likelihood is a product of conditional probabilities at each time point at which the event of interest occurs, we calculate the probability of the individuals experiencing the event at that time point, depending on the risk set at that time point. Maximizing the partial likelihood means solving a series of classification problems together. To do this, at each time point at which the event of interest is observed, we create a binary categorical variable representing the risk set and a covariate matrix containing the covariates for each sample in the risk set at that time. The binary categorical variable is created as much as the number of risk sets, and these created binary variables are placed side by side as columns and form the risk matrix. This risk matrix and the covariate matrix together form the prediction matrix of the model. We also create a binary vector that shows whether each individual in the risk set has experienced the event at the relevant time point. This binary vector is the outcome variable of the model. Finally, the prediction matrix and outcome variables created for each risk set are combined vertically, and this matrix creates the classification data matrix (Figure 2.13).

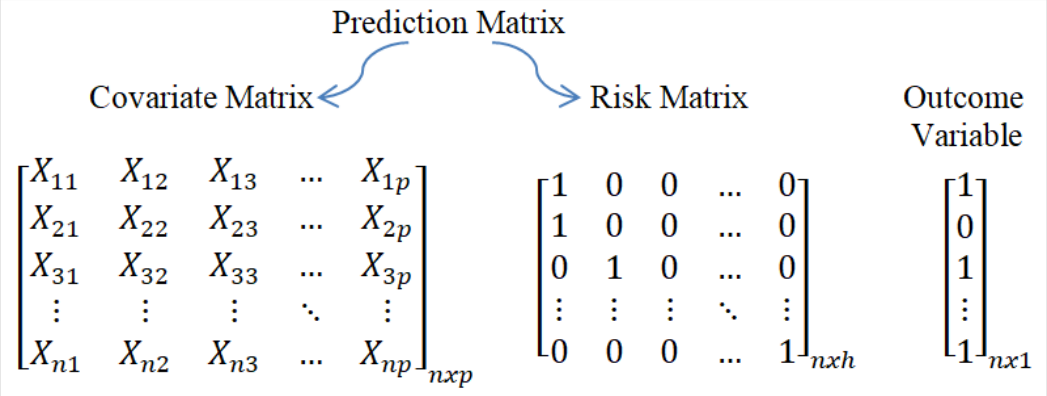


Figure 2.13. The data structure after the stacking idea.

An explanation of the stacking algorithm is given below to help you better understand it. An example survival data matrix with four individuals is shown in Table 2.5.

Table 2.5. An example survival data matrix.

Individuals	**Time**	**Status**	**Covariate Matrix of RNA-Seq**			
$\boldsymbol{i}$	$\boldsymbol{T_i}$	$\boldsymbol{\delta_i}$	$\boldsymbol{x_{i1}}$	$\boldsymbol{x_{i2}}$	**…**	$\boldsymbol{x_{ip}}$
1	9	1	4	10	…	0
2	8	0	5	14	…	2
3	6	1	7	18	…	8
4	10	1	3	9	…	5

First, the survival data are ordered from smallest to largest according to the time variable, shown in Table 2.6.

Table 2.6. Survival data matrix ordered by time.

Individuals	**Time**	**Status**	**Covariate Matrix of RNA-Seq**			
$\boldsymbol{i}$	$\boldsymbol{T_i}$	$\boldsymbol{\delta_i}$	$\boldsymbol{x_{i1}}$	$\boldsymbol{x_{i2}}$	**…**	$\boldsymbol{x_{ip}}$
3	6	1	7	18	…	8
2	8	0	5	14	…	2
1	9	1	4	10	…	0
4	10	1	3	9	…	5

Each individual's contribution to the partial likelihood is calculated in Table 2.7.

Cox proportional hazard model: $\lambda(t|x) = \lambda_0(t)\exp(x^T\beta)$

Partial likelihood:

$$L_{partial}(\beta) = \prod P(individual\ i\ event\ experience\ |R(T_i)\ risk\ set)$$
$$= \prod \frac{\exp(x_i^T \beta)}{\sum_{j \in R(T_j)} \exp(x_j^T \beta)}$$

There will be three risk sets since there are three uncensored individuals in the dataset. The first risk set represents the time from the beginning of the study to the sixth day and includes all individuals {1, 2, 3, 4} (Figure 2.14). The individuals in the first risk set are shown in Table 2.8, and the cumulative classification matrix created for this set after stacking is shown in Table 2.9.

Table 2.7. Contribution of each individual to partial likelihood.

Individuals	**Time**	**Sample at risk**	**Contribution of partial likelihood**
$\boldsymbol{i}$	$\boldsymbol{T_i}$	$\boldsymbol{R(T_i)}$	$[e^{\beta Z_i} / \sum_{j \in R(T_i)} e^{\beta Z_j}]^{\delta_i}$
3	6	{1,2,3,4}	$e^{\beta_0+7\beta_1+18\beta_2+\cdots+8\beta_p} / (e^{\beta_0+7\beta_1+18\beta_2+\cdots+8\beta_p} + \cdots + e^{\beta_0+3\beta_1+9\beta_2+\cdots+5\beta_p})$
2	8	{1,2,4}	1
1	9	{1,4}	$e^{\beta_0+4\beta_1+10\beta_2+\cdots+1\beta_p} / (e^{\beta_0+4\beta_1+10\beta_2+\cdots+1\beta_p} + e^{\beta_0+3\beta_1+9\beta_2+\cdots+5\beta_p})$
4	10	{1}	$e^{\beta_0+3\beta_1+9\beta_2+\cdots+5\beta_p} / e^{\beta_0+3\beta_1+9\beta_2+\cdots+5\beta_p} = 1$

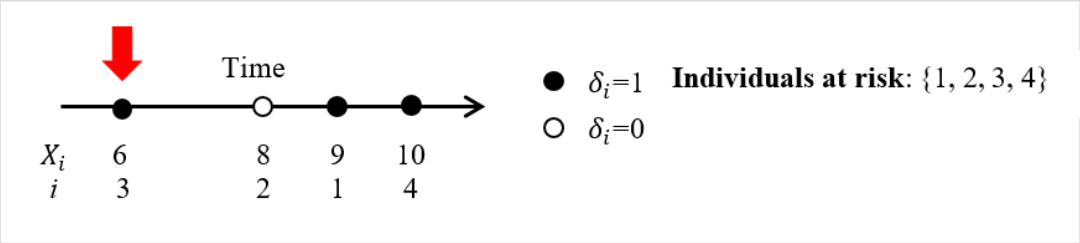


Figure 2.14. The figure presentation of risk set – 1

Table 2.8. Dataset of risk set – 1.

Individuals	**Time**	**Status**	**Covariate Matrix of RNA-Seq**			
$\boldsymbol{i}$	$\boldsymbol{T_i}$	$\boldsymbol{\delta_i}$	$\boldsymbol{x_{i1}}$	$\boldsymbol{x_{i2}}$	**…**	$\boldsymbol{x_{ip}}$
3	6	1	7	18	…	8
2	8	0	5	14	…	2
1	9	1	4	10	…	0
4	10	1	3	9	…	5

The second risk set represents the time from the beginning of the study to the ninth day, and there are two individuals {1, 4} in this cluster (Figure 2.15). The individuals found in the second risk set are shown in Table 2.10, and the cumulative classification matrix created for this set after stacking is in Table 2.11.

Table 2.9. Cumulative classification matrix for risk set – 1.

Prediction Matrix (X)							**Outcome Variable (Y)**
Covariate Matrix				**Risk Matrix**			
7	18	…	8	1	0	0	1
5	14	…	2	1	0	0	0
4	10	…	1	1	0	0	0
3	9	…	5	1	0	0	0

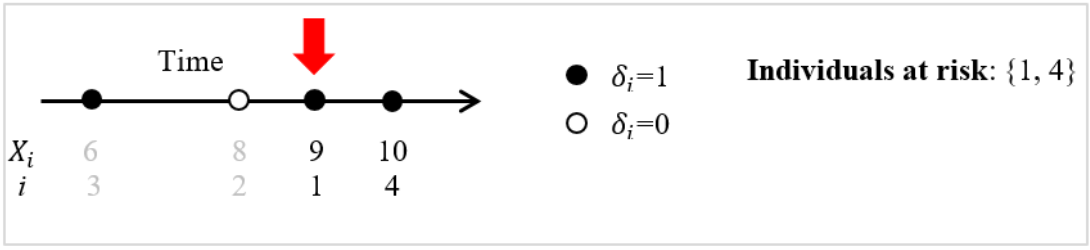


Figure 2.15. The figure presentation of risk set – 2.

Table 2.10. Dataset of risk set – 2.

Individuals	**Time**	**Status**	**Covariate Matrix of RNA-Seq**			
i	T_i	δ_i	x_{i1}	i	T_i	δ_i
1	9	1	4	1	9	1
4	10	1	3	4	10	1

Table 2.11. Cumulative classification matrix for risk set – 2.

Prediction Matrix (X)							**Outcome Variable (Y)**
Covariate Matrix				**Risk Matrix**			
7	18	…	8	1	0	0	1
5	14	…	2	1	0	0	0
4	10	…	1	1	0	0	0
3	9	…	5	1	0	0	0
4	10	…	1	0	1	0	1
3	9	…	5	0	1	0	0

The third risk set represents the time from the beginning of the study to the tenth day, and there is only one individual {4} in this cluster (Figure 2.16). The individual found

in the third risk set is shown in Table 2.12, and the cumulative classification matrix created for this set after stacking is in Table 2.13. After the stacking algorithm, the survival data matrix in Table 2.5 has been transformed into the binary classification data matrix in Table 2.13.

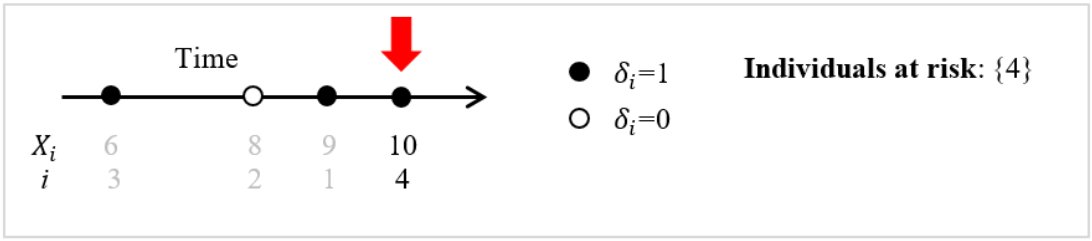


Figure 2.16. The figure presentation of risk set – 3.

Table 2.12. Dataset of risk set – 3.

Individuals	**Time**	**Status**	**Covariate Matrix of RNA-Seq**			
i	T_i	δ_i	x_{i1}	i	T_i	δ_i
4	10	1	3	4	10	1

The idea of stacking has been previously applied to logistic regression and given good results (206). The Cox proportional hazards model, which makes an estimation using the standard partial likelihood approach, is transformed into a classification problem using the stacking idea, depending on whether each individual experiences the event at each time point at which an event occurs (61). It has been shown that the predictions and results obtained from maximizing the partial probability in the Cox proportional hazards model are equivalent to the predictions and results made over the logistic regression parameters to the data converted by stacking. Also, the classification algorithms' performance after the stacking algorithm was higher than the Cox proportional hazards model.

Randomness can badly affect tree-based methods as the tree grows with randomly selected individuals through bootstrapping. Developing a single tree may yield different prediction results. Ensemble methods, on the other hand, treat each tree independently, employing a random set of explanatory variables at each node and ultimately considering all the results. The basic idea is that combining multiple survival tree estimators yields better predictions than a single independent tree. This

enhances the predictive performance compared to individual decision trees. Growing a full-size tree for each bootstrap sample also mitigates issues related to pruning and selection. Averaging the results of multiple trees helps reduce overfitting (180).

Table 2.13. Cumulative classification matrix for risk set – 3.

Prediction Matrix (X)							Outcome Variable (Y)	Time
Covariate Matrix				Risk Matrix				
7	18	…	8	1	0	0	1	6
5	14	…	2	1	0	0	0	8
4	10	…	1	1	0	0	0	9
3	9	…	5	1	0	0	0	10
4	10	…	1	0	1	0	1	9
3	9	…	5	0	1	0	0	10
3	9	…	5	0	0	1	1	10

This study will use the stacking idea for survival analysis of RNA-sequencing data. Thus, the stacking idea, shown to give better results than the classical Cox regression model when applied to clinical data, is expected to yield high-performance results when applied to RNA-seq high-dimensional data.

2.9. Priority-Lasso and IPF-Lasso

The prediction matrix, which is the result of applying the idea of stacking to RNA-seq survival data, contains two different types of data: the covariate matrix, which consists of continuous variables, and the risk matrix, which consists of binary variables. Priority-lasso and IPF-lasso algorithms allow the analysis of different types of variables in different blocks. Thus, it has been shown that the model's prediction performance increases (73,74).

Priority-Lasso algorithm puts variables in different blocks and gives these blocks different priority orders. Although the priority-Lasso algorithm has more characteristics, many of these characteristics are the same as the Lasso algorithm. Variable types are usually considered when creating blocks, such as continuous, discrete, binary, etc. Blocks can also be made according to variable contents, for example, a block with clinical variables, a block with genetic variables, etc. Blocks have a priority order. Accordingly, some blocks may have higher priority, while some blocks may not be of high priority. The researcher determines this priority level.

However, despite no absolute rule, high priority is given to blocks with easily accessible and low-cost variables. The prediction model is fit after applying Lasso regression as many as the number of blocks. It has been seen that the priority-lasso algorithm gives similar or better results than the standard Lasso algorithm in data such as multi-omics data where the variables in the data are of different types (73).

The IPF-Lasso algorithm was created from the necessity of applying different penalty terms to different data types in multi-omics datasets. This algorithm defined data types as modalities (data type = data modality). IPF-lasso applied different penalty factors to the data modalities in the process of combining the data in order to develop a more sparse estimation model for the data consisting of low and high dimensional variables. For this, the L1 penalized regression (LASSO) algorithm is used. IPF-LASSO performs better than the standard LASSO (74).

In this study, it is thought that the use of priority-Lasso and IPF-Lasso algorithms when analyzing the continuous and binary variables that occur after applying the stacking algorithms to the RNA-seq data can contribute positively to the prediction accuracy.

3. MATERIAL AND METHODS

In this section, the methodologies of the voomStackPrio and voomStackIPF approaches will be explained in detail (Figure 3.1). First, details about the structure of the RNA-seq survival data matrix will be provided. Second, the step-by-step process of developing new survival approaches will be explained. Then, the focus will be on elucidating how performance comparisons are conducted with other survival algorithms employed in the literature. Finally, the infrastructure of the MLSeqSurv R package utilized during the calculations will be mentioned.

3.1. Proposed RNA-Seq Survival Approaches

3.1.1. Notations

The data for survival analysis comprises two sets: covariates and outcome variables. Gene expression in RNA-seq data consists of raw counts, and these variables create covariates. Survival time and status of samples are the outcome variables. Assume that the covariates of RNA-seq gene expression data are a *n*x*p*-dimensional raw count data matrix representing *n* (*i*=1,2,..,*n*) samples and *p* (*g*=1,2,..,*p*) genes. This matrix is called *R*. i^{th} row of the *R* matrix is denoted by $r_{i.} = (R_{i1}, R_{i2}, \dots, R_{ip})$ and g^{th} column of the *R* matrix is denoted by $r_{.g} = (R_{1g}, R_{2g}, \dots, R_{ng})^T$. Accordingly, the read count of i^{th} sample and g^{th} gene is denoted by r_{ig}. The time variable, *T* ($T_i = T_1, T_2, \dots, T_n$), is a survival time. The status variable, δ ($\delta_i = \delta_1, \delta_2, \dots, \delta_n$), indicates whether there is censoring. If a sample has experienced the event of interest during the study period, the status is denoted by 1 (δ=1); or not, the status is denoted by 0 (δ=0). The survival data matrix is as in (Matrix 3.1).

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1p} \\ r_{21} & r_{22} & r_{23} & \dots & r_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & r_{n3} & \dots & r_{np} \end{bmatrix}_{nxp} \begin{bmatrix} T_1 & \delta_1 \\ T_2 & \delta_2 \\ \vdots & \vdots \\ T_n & \delta_n \end{bmatrix}_{nx2} \qquad (3.1)$$

We extracted the time variable (*T*) and state variable (δ) from the dataset and transposed the remaining matrix to initiate pre-processing. Normalization, transformation, and filtering steps were executed using Matrix 3.2.

$$\begin{bmatrix} r_{11} & r_{21} & r_{31} & \cdots & r_{n1} \\ r_{12} & r_{22} & r_{32} & \cdots & r_{n2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{1p} & r_{2p} & r_{3p} & \cdots & r_{np} \end{bmatrix}_{pxn} \quad (3.2)$$

To obtain the normalization factors for each gene, we then applied DESeq median normalization to the remaining dataset. Furthermore, low-expressed genes were identified from the *n*x*p*-dimensional raw RNA-seq matrix as described in Matrix 2.1. Following DESeq normalization and logCPM transformation, these identified genes will be excluded from the dataset.

3.1.2. DESeq Median Normalization

In a comprehensive study that compared various normalization methods, TMM and DESeq emerged as the best-performing methods (133). Therefore, we have chosen to employ the DESeq median normalization method in this study.

The geometric mean over all samples is used to calculate the pseudo-reference value for each gene. Specifically, the pseudo-reference value is computed as follows for the *g* gene in the dataset with *p* genes and *n* sample.

$$s_g = \sqrt[n]{r_{g1} r_{g2} \dots r_{gn}} = (\prod_{i=1}^{n} r_{gi})^{1/n} \; g=1,2,\dots,p \quad (3.3)$$

These geometric mean values calculated for each gene generate a new sample called the pseudo-reference sample. Subsequently, each gene value in every sample is divided by the corresponding pseudo-sample value of that gene. Then, the median values for each sample are computed.

$$d_i = median_g \frac{r_{gi}}{(\prod_{i=1}^{n} r_{gi})^{1/n}} = median_g \frac{r_{gi}}{s_g} \quad (3.4)$$

These median values calculated for each sample are the normalization factors (size factors). The objective of this normalization step was not to derive normalized values for each count but solely to compute normalization factors for each sample. These normalization factors will be applied in the subsequent step to obtain logCPM values.

3.1.3. voom Transformation

The voom transformation yields two distinct outputs: logCPM values and sample weights.

logCPM Values

The raw RNA-seq count data matrix is denoted as r_{gi}. To obtain logCPM values for the data set, it is necessary to divide the count data by the normalization factors (d values in Equation 3.4) and multiply by 1 million. The formula for this calculation is presented below

$$x_{gi} = log_2\left(\frac{r_{gi}+0.5}{d_i+1} * 10^6\right) \qquad (3.5)$$

d_i are the values calculated in the normalization step for each sample. 0.5 is added to each count value to prevent the logarithm from being 0. In addition, 1 was added to d_i to obtain the equality of $0 \leq \frac{r_{gi}+0.5}{d_i+1} \leq 1$.

Following the normalization steps, logCPM value generation, and low-expressed gene filtering, matrices for the dataset is presented in (Matrix 3.6).

$$\begin{bmatrix} x_{11} & x_{21} & \dots & x_{n1} \\ x_{12} & x_{22} & \dots & x_{n2} \\ x_{13} & x_{23} & \dots & x_{n3} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1p'} & x_{2p'} & \dots & x_{np'} \end{bmatrix}_{p' x n} \qquad (3.6)$$

voom Transformation for Observational Weights

In the second stage of the voom transformation, sample weights are computed to take advantage of the sample-specific weighting approach (151).

A linear model fits the data following logCPM transformation. Specifically, the model assumes that $E(x_{gi}) = \mu_{gi} = a_i^T \beta_g$, where a_i is a vector of covariates and β_g is a vector of unknown coefficients (69).

The linear model $x_g = D\beta_g + \varepsilon_g$ and $E(x_g) = D\beta_g$ is assumed for each gene. $x_g = (x_{g1}, \dots, x_{gn})^T$ vector of logCPM values for the gene g; D is the design matrix and $\beta_g = (\beta_{g1}, \dots, \beta_{gK})^T$ is the vector of the regression coefficients for the gene g.

ε_g is the error term an $E(\varepsilon_g) = 0$. This yields regression coefficient estimates $\hat{\beta}_g$, fitted values $\hat{\mu}_{gi} = a_i^T \hat{\beta}_g$, and residual standard deviations s_g (69).

Suppose the expected value of a count is $E(r) = \lambda$ and $var(r) = \lambda + \phi\lambda^2$. Here ϕ is a dispersion parameter. If r is large enough, the logCPM value of the observation; $x \approx log_2(r) + log_2(d) + 6log_2(10)$. Since d will behave like a constant, it becomes $var(x) \approx var(log_2(r))$. Based on the delta rule and Taylor's theorem (207), if λ is large, $log_2(r) \approx \lambda + (r - \lambda)/\lambda$ from $var(x) \approx \frac{var(r)}{\lambda^2} = \frac{1}{\lambda} + \phi$.

The x_{gi} values, representing the logCPM values calculated in the 'logCPM Values' step for each gene, are subjected to fitting based on the aforementioned linear model. The calculation of the mean $\bar{x}_g$ for each gene is carried out using the $\tilde{r} = \bar{x}_g log_2(\tilde{d}) - log_2(10^6)$.

μ_{gi} is estimated with $\hat{\lambda}_{gi} = \hat{\mu}_{g,} + log_2(d_i + 1.0) - log_2(10^6)$ ($\hat{\mu}^*_{gi} = E(x^*_{gi})$) by fitting a LOWESS curve (208). The piecewise linear function $lo(\hat{\lambda}_{gi})$ defined by the LOWESS curve is the estimated square root standard deviation of the mean log counts $\tilde{r}$ ($s_g^{(\frac{1}{2})}$).

The voom precision weights are inverse variances of $w_{gi} = lo(\hat{\lambda}_{gi})^{-4}$. For the dataset, x_{gi} is the logCPM values and w_{gi} is the associated weights for each counts.

The design matrix *D* denotes the experimental design and selects the regression coefficients and parameterization, presenting the logCPM variability among the RNA sources in the experiment. This model assumes $var(x_{gi}) = \sigma_g^2/w_{gi}$ for gene g in sample i, using an observational level weight w_{gi} derived from the voom model as found above and an unknown factor σ_g^2.

In addition to gene-dependent variance factors (σ_g^2) that account for variations among genes, there are sample-dependent variance factors (σ_{gi}^2) reflecting potential differences in quality across all or most genes within a given sample (151). This can result in an increase or decrease in their variability, as illustrated below (70).

$$var(x_{gi}) = \frac{\sigma_{gi}^2}{w_{gi}}$$

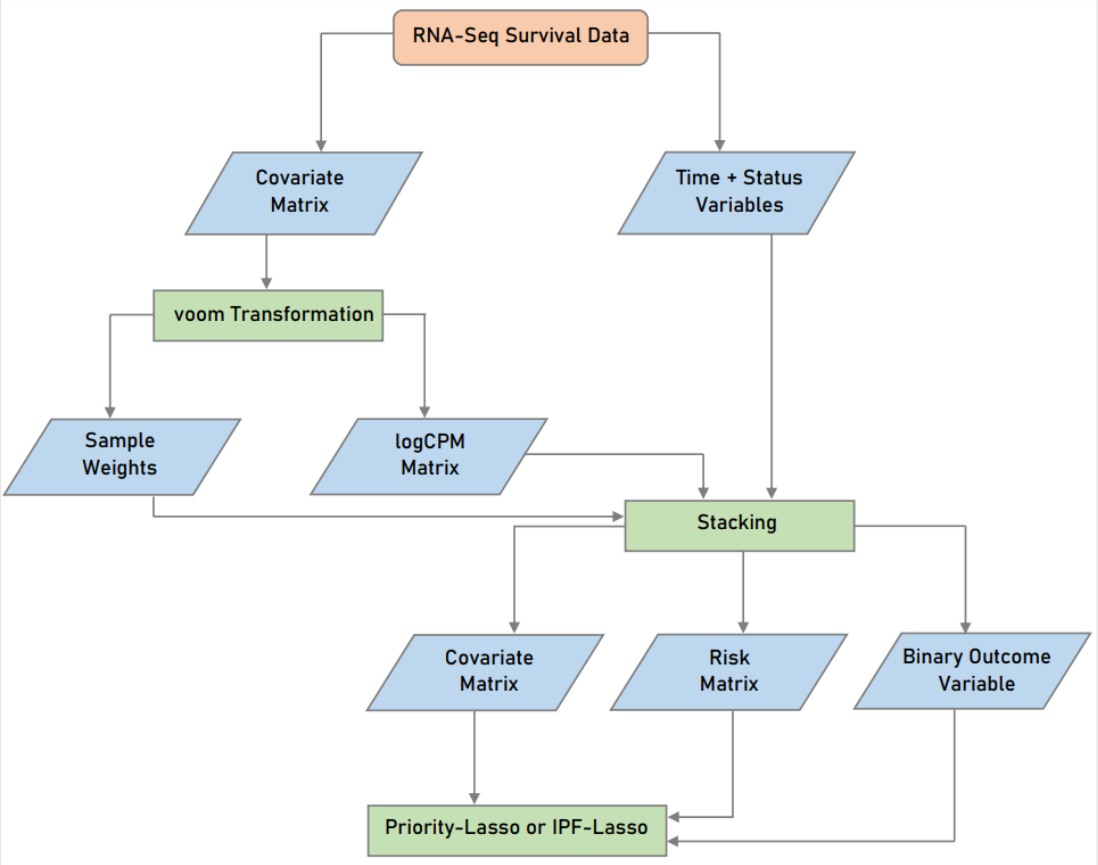


Figure 3.1. A flowchart of the steps of voomStackPrio and voomStackIPF algorithms.

The most straightforward log-linear model, ensuring that variability is multiplicatively dependent on sample quality, is expressed as $log\sigma_{gi}^2 = \delta_g + \gamma_i$. The constraint $\sum_{i=1}^{n} \gamma_i = 0$ gives $\sigma_g^2 = \exp(\delta_g)$ for the variance factors by gene and γ_i represents the relative variability of each sample. A given sample i is of relatively better-than-mean quality if $\gamma_i < 0$, or of poorer-than-mean quality if $\gamma_i > 0$. Linear modeling incorporates 'voom precision weights' for each observation, combined with sample-specific weights, as described in $w_{gi}^* = w_{gi}/exp\hat{\gamma}_i$, where w_{gi} represents the observational voom weights (151).

The weights generated for each sample in the dataset are as follows (Equation 3.7). These will be the sample weights in the priority-Lasso and IPF-Lasso models applied in the 'voomStackPrio and voomStackIPF Models' step.

$$w_i = (w_1, w_2, \dots, w_n) \qquad (3.7)$$

Before proceeding to the stacking algorithm, we applied variance filtering and feature selection. Following variance filtering, 2000 genes were selected. Subsequently, after applying feature selection, p'' features were chosen. Then, the matrix was transposed once more to apply the stacking algorithm. Also, the time variable (T) and state variable (δ), previously removed from the dataset to apply of pre-processing steps, were reintegrated (Matrix 3.8).

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1p''} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2p''} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{np''} \end{bmatrix}_{nxp''} \begin{bmatrix} T_1 & \delta_1 \\ T_2 & \delta_2 \\ \vdots & \vdots \\ T_n & \delta_n \end{bmatrix}_{nx2} \quad (3.8)$$

3.1.4. Stacking for Classification

This step involved a conversion process, utilizing the stacking approach, to transform RNA-seq survival data into classification data with a binary outcome.

Let's consider *h* as the number of samples that experience the event in the data set. Following the stacking algorithm, there will be *h* risk sets, corresponding to the number of columns in the risk matrix, denoting the risk set as *S*. In each risk set at time *t*, some samples either experienced the event at time *t*, experienced the event after time *t*, or were censored after time *t*. $S(t) = \{subject\ i\ |t_i \geq t\}$. Samples that are part of the risk set at time *t* are assigned a value of 1, and samples outside of it are assigned a value of 0. This information is displayed in the specific column of the risk set at that time. The $\tilde{X}$ covariate matrix is constructed as $\tilde{X}(S(i))$, where the covariate number *S(i)* is associated with each sample. It has dimensions |*S(i)*|x*p''* for *p''* features. Let $\tilde{y}$ represent the prediction of the binary outcome variable we will create for the classification transformation. For the $\tilde{y}$ binary outcome variable, samples that experience the event at the time associated with each risk set are assigned the value 1, while others receive the value 0. Thus, $\tilde{y}(S(i))$ values are defined. The covariate matrix, risk matrix, and binary outcome are created as many times as the risk sets generated for each uncensored sample. They are then added vertically, one after the other. This conversion results in $(\tilde{X}, \tilde{y})$ data, suitable for applying classification algorithms to the survival data.

After implementing the stacking algorithm, three components will be obtained (*i*) the covariate matrix, (*ii*) the risk matrix, and (*iii*) the outcome variable, as shown in (Matrix 3.9). The data matrix $x_{np''}$ in Matrix 3.8 is initially ordered based on time, arranged from smallest to largest. This covariate matrix is then vertically expanded by stacking. Hence, the covariate matrix consists of p'' columns and n' rows denoted by $covariate_{n'xp''}$. Also, a risk matrix is obtained. The risk matrix expanded horizontally and vertically per the number of individuals experiencing the event. As the number of risk sets corresponds to the number of samples experiencing the event, the columns of the risk matrix at risk are equivalent to the number of samples in which the event occurred (h). So, an n'xh-dimensional risk matrix, denoted as $risk_{n'xh}$, is obtained. There is a generated outcome variable for every risk set that exists at the event time point. The variables in question designate samples that encounter the event at that particular time point as 1, and samples that do not are defined as 0. As denoted by $x_{n'(p''+h)}$, the input matrix ($\tilde{X}$) has now transformed into an n'x($p''+h$)-dimensional matrix, incorporating both the covariate and risk matrices. An n'x1-dimensional outcome variable ($\tilde{y}$), denoted as $outcome_{n'x1}$, is obtained.

Covariate Matrix Risk Matrix Outcome Variable

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1p''} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2p''} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3p''} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n'1} & x_{n'2} & x_{n'3} & \dots & x_{n'p''} \end{bmatrix}_{n'xp''} \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}_{n'xh} \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{n'x1} \quad (3.9)$$

3.1.5. voomStackPrio and voomStackLasso Models

We have two different types of data in our dataset: a binary risk matrix and a continuous covariate matrix. In contrast to conventional classification algorithms, our method relies on modeling that takes into account the particular kinds of data related to the variables—a tactic that has been shown to produce predictions that are more accurate. We employed the priority-Lasso and IPF-Lasso algorithms to analyze diverse variable types in multi-omics data organized into blocks. In the

subsequent stage, two new approaches–voomStackPrio and voomStackIPF– were developed to analyze RNA-seq survival data.

Lasso and weighted Lasso regression model

Assume that x_{ig} represents the observed value of the gth variable for the i^{th} sample, where g=1, 2,…, $(p''+h)$, i=1, 2,…, n'. The outcome of sample i is denoted as y_i. In the classical Lasso method, estimating regression coefficients $\beta_1, \dots, \beta_{(p''+h)}$ for the $(p''+h)$ variables involves minimizing the following objective function with respect to $\beta_1, \dots, \beta_{(p''+h)}$.

$$\sum_{i=1}^{n'} (y_i - \sum_{g=1}^{(p''+h)} x_{ig}\beta_g)^2 + \lambda \sum_{g=1}^{(p''+h)} |\beta_g|$$

In this context, λ represents the penalty parameter, which controls the degree of shrinkage applied to the regression coefficient estimates. By tuning the value of λ, the Lasso method regularizes these estimates, preventing overfitting and enhancing the model's capacity to generalize effectively to new data. The optimal λ value is typically accomplished through cross-validation, a statistical technique that evaluates the model's performance on an independent dataset.

While sample weights are often ignored in many Lasso regression models, a constructive approach to address this omission is to incorporate sample weights into the Lasso regression model. By assigning distinct weights to individual observations, the weighted Lasso regression model can attribute greater significance to specific observations, potentially enhancing the precision of the estimates. The goal of the weighted Lasso regression is to minimize the following objective function concerning $\beta_1, \dots, \beta_{(p''+h)}$

$$\sum_{i=1}^{n'} w_i (y_i - \sum_{g=1}^{(p''+h)} x_{ig}\beta_g)^2 + \lambda \sum_{g=1}^{(p''+h)} |\beta_g|$$

where w_i represents the weight assigned to the i^{th} sample. This study will utilize sample weights derived from the voom transformation in Equation 3.7 applied to RNA-seq data in block-based Lasso algorithms.

voomStackPrio

The priority-Lasso algorithm was applied to the n'x$(p'' + h)$-dimensional prediction matrix *X*, including the covariate and risk matrices (Matrix 3.9). The outcome variable *y*, a n'x*1*-dimensional vector, was obtained after the 'Stacking for Classification' step (Matrix 3.9). To streamline this process, we organized variables into two blocks based on their types: the continuous variables block for those in the covariate matrix and the binary variables block for those in the risk matrix. Notably, priority was given to the binary variable block. This strategic decision aligns with the principle of the priority-Lasso algorithm, where the highest-priority block plays a crucial role in explaining variability. Variables in lower-priority blocks are considered only if they contribute to variances not already explained by higher-priority blocks. Consequently, prioritizing the block with binary risk matrix variables, less complex than RNA-seq continuous variables, is considered more suitable.

The variables in the two blocks for the i^{th} sample can be represented as follows

$$x_{i1}^{(m)}, \dots, x_{i(p''+h)_m}^{(m)} \quad , \quad i = 1, \dots, n' \quad m = 1,2$$

The number of blocks is denoted by *m*, and the number of variables in the block is denoted by $(p'' + h)_m$. The regression coefficients of variable *j* are shown as follows

$$\beta_1^{(m)}, \dots, \beta_{(p''+h)_m}^{(m)} \quad , \quad g = 1, \dots, (p'' + h)_m$$

The vector π=(π_1, π_2) denotes the blocks descending order of priority. π_1 represents the first (highest-priority) block, and π_2 represents the second (lower-priority) block.

Initially, a Lasso model was applied to the high-priority binary variable block. The goal of this step is primarily to capture the variability in the outcome variable using variables within this block. The first block consists of *h* variables, and the variables for the i^{th} sample are depicted as follows.

$$x_{i1}^{(1)}, \dots, x_{ih}^{(1)} \quad , i = 1, \dots, n'$$

The h binary variables in block π_1 are employed to fit the initial Lasso regression model. The coefficients $\beta_1^{(\pi_1)}, \ldots, \beta_{h_{\pi_1}}^{(\pi_1)}$ are estimated by minimizing the following formula

$$\sum_{i=1}^{n'} w_i \left(y_i - \sum_{g=1}^{h_{\pi_1}} x_{ig}^{(\pi_1)} \beta_g^{(\pi_1)} \right)^2 + \lambda^{(\pi_1)} \sum_{g=1}^{h_{\pi_1}} |\beta_g^{(\pi_1)}|$$

w_i represents the weights assigned to the samples, derived from the voom transformation.

The variables in the second block account for the remaining variability in the outcome variable after explaining the portion addressed by the variables in the π_1 block. The linear score obtained from the Lasso model fitted in the first block serves as an offset, and a second Lasso model is fitted to the second block, which consists of continuous variables. This involves fitting the second Lasso model to the residuals from the first Lasso model without incorporating the offset, using the covariates in the π_2 block. The linear predictor to be employed as an offset in the second Lasso model is fitted in the first Lasso model as follows

$$\hat{\eta}_{1,i}(\pi) = \hat{\beta}_1^{(\pi_1)} x_{i1}^{(\pi_1)} + \cdots + \hat{\beta}_{h_{\pi_1}}^{(\pi_1)} x_{ih_{\pi_1}}^{(\pi_1)}$$

However, the linear estimation of $\hat{\eta}_{1,i}(\pi)$ can be over-optimistic and may result in underestimating the π_2 block. This is because y_i is part of the data used to estimate the β coefficients employed in calculating this linear estimate. To address this issue, cross-validation was employed to estimate the offset of $\hat{\eta}_{1,i}(\pi)$. The dataset, X, was divided into K roughly equal-sized portions, denoted as $k=1,\ldots,K$. The coefficients $\hat{\beta}_{X/X_k,1}^{(\pi_1)}, \ldots, \hat{\beta}_{X/X_k,h_{\pi_1}}^{(\pi_1)}$ were estimated, and cross-validated offsets are calculated as follows

$$\hat{\eta}_{1,i}(\pi)_{CV} = \hat{\beta}_{X/X_k,1}^{(\pi_1)} x_{i1}^{(\pi_1)} + \cdots + \hat{\beta}_{X/X_k,h_{\pi_1}}^{(\pi_1)} x_{ih_{\pi_1}}^{(\pi_1)}$$

The second block (π_2) consists of p'' variables, and the variables for the i^{th} sample are shown as follows

$$x_{i1}^{(2)}, \ldots, x_{ip''}^{(2)} \qquad , i = 1, \ldots, n'$$

The coefficients $\beta_1^{(\pi_2)}, \ldots, \beta_{p''_{\pi_2}}^{(\pi_2)}$ for the second block (π_2) are estimated by minimizing the following formula

$$\sum_{i=1}^{n'} w_i \left(y_i - \hat{\eta}_{1,i}(\pi)_{CV} \sum_{g=1}^{p''_{\pi_2}} x_{ig}^{(\pi_2)} \beta_g^{(\pi_2)} \right)^2 + \lambda^{(\pi_2)} \sum_{g=1}^{p''_{\pi_2}} |\beta_g^{(\pi_2)}|$$

voomStackIPF

As described in the priority-Lasso algorithm, the n'x$(p'' + h)$-dimensional *X* prediction matrix, obtained at the conclusion of the 'Stacking for Classification' step, encompasses continuous RNA-seq covariate variables and binary variables (Matrix 3.9). Given the distinct nature of these variables, the data type (or the number of data modalities) is designated as two. Let the modality number be denoted by *m* (*m*=1, 2). Variables in each modality are defined as follows

$$x_{i1}^{(m)}, \ldots, x_{i(p''+h)_m}^{(m)} \quad , i = 1, \ldots, n' \quad m = 1,2$$

$(p'' + h)_m$ represents the number of variables in modality *m*. The g^{th} variable is denoted by $x_g^{(m)}$, and its corresponding coefficient is represented by $\beta_g^{(m)}$. In the IPF-lasso algorithm, a weighted sum of the norms of the coefficients vector for each modality is employed as a penalty term. To estimate the coefficients, the following formula is minimized

$$\sum_{i=1}^{n'} w_i \left(y_i - \sum_{m=1}^{2} \sum_{g=1}^{(p''+h)_m} x_{ig}^{(m)} \beta_g^{(m)} \right)^2 + \sum_{m=1}^{2} \lambda_m ||\beta_g^{(m)}||_1$$

λ_m represents the penalty for the variables in modality *m*. The first modality (λ_1 penalty) was treated as the reference modality, and the penalty factor for modality *m* was expressed as λ_m/λ_1.

In this scenario, the penalty factors were defined as $\lambda_1, \lambda_2/\lambda_1$. Cross-validation was applied to improve prediction performance. The various candidate vectors of penalty factors are denoted as *C* and listed below

$$n'^{(c)} = (1, \lambda_2/\lambda_1)^T \qquad c = 1, \ldots, C$$

Cross-validation was performed using a performance metric such as AUC to determine the optimal λ_1.

3.2. Performance Evaluation

3.2.1. Transformation of Test Data into Classification Data

A test matrix containing survival data was employed to evaluate the newly developed algorithms for the survival analysis of RNA-seq data in '3.1. Proposed RNA-Seq Survival Approaches' step. All pre-processing and stacking algorithm steps applied to the training dataset were also executed on the test set, continuing until block-based Lasso models were applied. The parameters used during normalization for the training set were also consistently applied to the test set. However, the test set was normalized independently of the training data, ensuring that both the training and test sets were on the same scale and exhibited homoscedasticity. Low-expressed genes excluded from the training set were also excluded from the test set. The parameters used in the training sets during the voom transformation were also used in the test sets, similar to the normalization step. The 2000 genes with the highest variance in the training set were also selected in the test set. In the feature selection step, the variables selected in the training set were also selected in the test set.

For the stacking algorithm, the same methodology employed to construct the risk set for the training dataset was applied to generate risk set variables for the test set. Initially, the time values of individuals experiencing the event in the training set were arranged in ascending order. Subsequently, a risk matrix vector was created for the relevant individual based on the range in which the time variable of each individual in the test set falls in this order. To illustrate, suppose the time value of an individual in the test set is 1578. Assuming there are ten risk set variables in the training set with corresponding time variables (100, 300, 590, 1080, 1432, 1602, 1845, 1936, 2010, 2036) belonging to individuals who experienced the event in that set. Time 1578 aligns with the 5th interval in this list. Consequently, the risk set values for this test set observation would be (0, 0, 0, 0, 1, 0, 0, 0, 0, 0). The computation of risk set values is

performed individually for each person in the risk set, and collectively, these values constitute the risk set for the test set, following the approach used in the training set.

3.2.2. RNA-Seq Datasets

Real RNA-seq survival data were used in this study, which concentrated on TCGA data that included RNA-seq data for 12 different cancer types. Data on read counts were obtained from the TCGA data portal (https://portal.gdc.cancer.gov/), and the R program's **TCGAbiolinks** package (209) was used to carry out the download operation. Each dataset comprises 60660 genes, including 19938 protein-coding genes and 40722 non-coding genes. However, only protein-coding genes were considered for this study. For cancer types other than LAML, the sample type "Primary Tumor" was selected, while for LAML, individuals with the sample type "Primary Blood Derived Cancer - Peripheral Blood" were included in the analysis. The overall survival time and status data associated with RNA-seq count data were extracted from the TCGA Clinical Data Resource, resulting from a comprehensive study involving 11,000 cancer patients across 33 different cancer types in TCGA (210). The characteristics of the datasets are summarized in Table 3.1 and Table 3.2.

Table 3.1. RNA-Seq Datasets.

Data Code	Cancer Type	Sample Size (*n*)	Zero/Null Time Filtering	Censoring Rate (0/1)
ACC	Adrenocortical Carcinoma	79	79	51/28
CESC	Cervical Squamous Cell Carcinoma and Endocervical Adenocarcinoma	304	291	220/71
ESCA	Esophageal Carcinoma	184	184	107/77
GBM	Glioblastoma Multiforme	155	154	32/122
KIRC	Kidney Renal Clear Cell Carcinoma	529	527	352/175
KIRP	Kidney Renal Papillary Cell Carcinoma	290	287	243/44
LAML	Acute Myeloid Leukemia	151	130	52/78
LGG	Brain Lower Grade Glioma	516	511	386/125
MESO	Mesothelioma	87	85	12/73
PAAD	Pancreatic Adenocarcinoma	178	177	84/93
SARC	Sarcoma	259	259	161/98
UVM	Uveal Melanoma	80	80	57/23

3.2.3. Evaluation Process

The procedures applied to the real datasets are detailed in the following step-by-step manner. Additionally, a visual representation of these steps is presented in Figure 3.2 via a flowchart.

Table 3.2. Patient Characteristics.

Data Code	Age	Gender (Female)	Overall Survival Time (days)	Censoring Rate (=0)
ACC	46.70±15.77	48 (60.76)	1194 (662-2056)	51 (64.6)
CESC	48.09±13.81	100 (100.00)	699 (410-1345)	220 (75.6)
ESCA	62.45±11.93	26 (14.13)	396.50 (231.25-675.75)	107 (58.2)
GBM	59.69±13.60	57 (37.01)	350.00 (153.00-535.50)	30 (19.5)
KIRC	60.56±12.17	186 (35.23)	1217.00 (551.00-1929.00)	352 (66.8)
KIRP	61.04±13.00	76 (26.39)	771.00 (428.00-1508.00)	243 (84.7)
LAML	53.52±16.32	59 (45.38)	366.00 (184.00-861.00)	53 (40.8)
LGG	43.02±13.36	228 (44.62)	678.00 (405.00-1227.00)	386 (75.5)
MESO	63.05±9.83	16 (18.82)	527.00 (258.00-852.00)	12 (14.0.)
PAAD	64.52±10.93	80 (45.20)	466.00 (277.50-680.00)	84 (47.5)
SARC	60.71±14.59	141 (54.44)	947.00 (485.00-1585.00)	161 (62.2)
UVM	61.65±13.95	35 (43.75)	784.00 (433.50-1182.50)	57 (71.3)

Splitting datasets: In the first step of the process, the data is split into two: the training set and the test set. The training set is designated for developing the voomStackPrio and voomStackIPF approaches, while the test set is reserved for evaluating the trained model. The RNA-seq survival data matrix with *n* samples, as illustrated in (Matrix 3.1), was randomly split into 70% for the training set and 30% for the test set. The status variables within the training and test sets are categorized into two groups, denoted by values 0 and 1. Because these groups appeared in the status variable a certain number of times, they were divided equally between the training and test sets during the splitting step to prevent bias. For example, in the training set, there are 105 samples with status=1 and 35 samples with status=0, while in the test set, there are 45 samples with status=1 and 15 samples with status=0. It is assumed that the

training set comprises n_1 samples (rows), and the test set comprises n_2 samples (rows), where $n_1 + n_2 = n$. The matrices representing the training and test sets are depicted in (Matrix 3.10) and (Matrix 3.11).

$$\begin{bmatrix} rTrain_{11} & rTrain_{12} & rTrain_{13} & \dots & rTrain_{1p} \\ rTrain_{21} & rTrain_{22} & rTrain_{23} & \dots & rTrain_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ rTrain_{n_1 1} & rTrain_{n_1 2} & rTrain_{n_1 3} & \dots & rTrain_{n_1 p} \end{bmatrix}_{n_1 x p} \begin{bmatrix} T_1 & \delta_1 \\ T_2 & \delta_2 \\ \vdots & \vdots \\ T_{n_1} & \delta_{n_1} \end{bmatrix}_{n_1 x 2} \quad (3.10)$$

$$\begin{bmatrix} rTest_{11} & rTest_{12} & rTest_{13} & \dots & rTest_{1p} \\ rTest_{21} & rTest_{22} & rTest_{23} & \dots & rTest_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ rTest_{n_2 1} & rTest_{n_2 2} & rTest_{n_2 3} & \dots & rTest_{n_2 p} \end{bmatrix}_{n_2 x p} \begin{bmatrix} T_1 & \delta_1 \\ T_2 & \delta_2 \\ \vdots & \vdots \\ T_{n_2} & \delta_{n_2} \end{bmatrix}_{n_2 x 2} \quad (3.11)$$

The time variable (*T*) and the status variable (δ) were taken out of the training and test matrices in the dataset during the next pre-processing steps. Additionally, the transposes of both matrices were obtained. This partitioning process was done using the `partition()` function within the **mlr3** package (211).

Normalization: The next step is to normalize the data for both training and test sets after they have been divided. The DESeq median ratio algorithm was utilized in this normalization process to obtain normalized values. The normalization process involves leveraging the `estimateSizeFactors()` and `estimateDispersions()` functions from the **DESeq2** package (68), as well as the `calcNormFactors()` functions from the **edgeR** package (97). Consistently applied to the test set were the same parameters that were used for training set normalization. However, the test set was normalized independently of the training data, ensuring that both the training and test sets were on the same scale and exhibited homoscedasticity.

Filtering low-expressed genes: The approach of Chen et al. (212) is applied to remove genes that are unexpressed or low-expressed (unchanging or low-variability) across all the libraries using the function `filterByExpr()` from the **edgeR** package (97). This function tries to keep genes with at least minimum count reads in a worthwhile number of samples. According to this approach, we keep genes with CPM above the minimum count (default *k*=10) in a minimum proportion of samples in the minimum

group sample size (*n*). The minimum proportion is greater than 70% of the smallest group size as default.

Removing the filtered unexpressed or low-expressed genes from the data before the normalization step may change the data's original structure. Therefore, first of all, the genes to be filtered were identified in the training set before normalization. Post normalization and logCPM transformation on the training set encompassing all genes identified were excluded. The excluded genes from the training set were also removed from the test set.

Transformation: In the methods utilized for comparing model performance, we applied the variance stabilizing transformation (vst) to the normalized values. For other algorithms to compare, this transformation was achieved using the `varianceStabilizingTransformation()` function within the **DESeq2** package (68). For our newly developed algorithms, voomStackPrio and voomStackIPF, we implemented the voom transformation on the normalized values. We reorganized the code of the `voom()`, `CalcNormFactors()`, `arrayWeights()`, and `voomWithqualityweights()` functions in the **limma** (213) and **edgeR** (97) packages. The parameters used in the training sets were also utilized in the test sets, similar to the normalization step.

Variance filtering: To improve analysis accuracy, more informative genes were prioritized and the genes were sorted in descending order according to their coefficients of variation. The studies were conducted using the top 2000 genes from this ordered list. However, considering potential variations in the coefficient of variation values after transformation, we identified the initial 2000 genes for analysis before the transformation step. Following the transformation, a variance filtering process was applied. This procedure was implemented in the training set, and subsequently, the genes filtered in the training set were also filtered in the test set.

Feature selection: Two distinct feature selection methods were employed to compare model performance. The first method involved model-based feature selection, which is implemented differently for each survival model in the **mlr3fselect** package (214). Resampling techniques are used by the algorithms in the **mlr3fselect** package (214) to assess prediction performance and choose feature subsets. For feature selection, resampling was conducted using 5 repeats of 5-fold cross-validation, with the

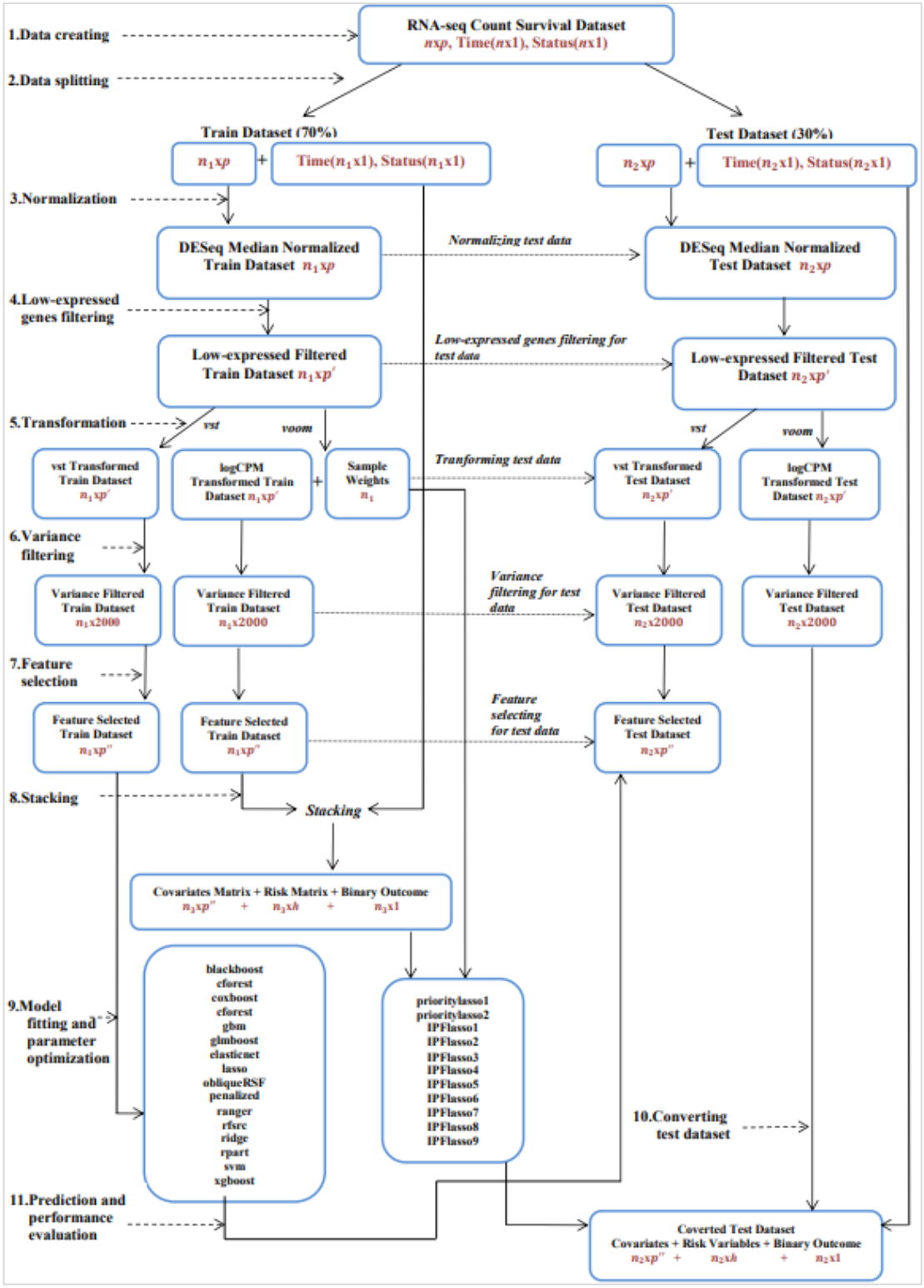


Figure 3.2. Workflow of evaluation process.

performance measure set to ‘c-index’, and feature selection was completed using the ‘random search’ algorithm. The features selected in the training set were applied to the test set.

The second method is the Boruta algorithm, which is also used in the feature selection for our newly developed lasso-based methods, voomStackPrio and voomStackIPF. In the Boruta feature selection from **Boruta** package (152) process, features labeled as both ‘important’ and ‘tentative’ (closely resembling the best shadow features) were retained in the dataset, while variables labeled as ‘unimportant’ were subsequently removed. The features in the training dataset are also kept in the test dataset. Detailed information regarding the number and names of the variables used for each model can be found in the Appendix Files.

Hence, the pre-processing steps for RNA-seq data are now complete. We transposed the training and test datasets to facilitate subsequent steps and reintroduced each sample's time and status variables.

Stacking: We developed a function that followed the steps of the stacking algorithm to convert the training set from a survival data matrix into a classification data matrix. The survival test dataset was similarly transformed into a classification dataset by leveraging the risk sets generated during the application of the stacking algorithm to the training set.

Model fitting and parameter optimization: Multiple models with different parameters were developed for the new approaches. The specific parameters used for voomStackPrio can be found in Table 3.3, while those for voomStackIPF are listed in Table 3.4. The ‘weights’ parameter in these tables denotes sample weights obtained after the voom transformation. Given that our outcome variable *y* is binary in voomStackPrio, the ‘family’ parameter is selected as ‘binomial’, and the ‘type.measure’ parameter is set to ‘auc’. The first block is penalized in two voomStackPrio models. The ‘lambda.type’ parameter determines the lambda value used in predictions. ‘lambda.min’ provides the lambda with minimum cross-validated errors, and ‘lambda.1se’ gives the largest lambda value within one standard error of the minimum. The ‘standardized’ parameter determines whether estimates would be standardized or not. For voomStackIPF, the ‘alpha’ parameter plays a pivotal role. When set to 1, it applies an L1-penalty (lasso), and when set to 0, an L2-penalty (ridge)

is used. Each model for voomStackPrio and voomStackIPF was run 30 times, employing 10 repeats of 5-fold cross-validation.

The performance of the newly created voomStackPrio and voomStackIPF models was compared to other survival analysis methods found in the literature.

These methods were categorized into four primary groups: (i) penalized Cox regression methods, (ii) boosted survival methods, (iii) random survival forests, and (iv) support vector machines. Hyperparameters for these machine-learning algorithms were carefully selected to optimize model performance. The tuning of these hyperparameters was carried out automatically using a 5-fold 10-repeated cross-validation process. Importantly, optimal hyperparameters were chosen from different ranges for different models, and the specific tuning parameters for each model are detailed in Table 3.5. To ensure robustness and reliability, each model underwent 25 iterations, randomly selecting 30 distinct training and test datasets.

The steps applied to the algorithms for comparison are detailed below.

blackboost: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.blackboost', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.blackboost’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

coxboost: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.coxboost', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.coxboost’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

gbm: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.gbm', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the 'surv.gbm' function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

glmboost: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.glmboost', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the 'surv.glmboost' function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

xgboost_gbtree: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.xgboost', 'booster = "gbtree"', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the 'surv.xgboost' function and 'booster = "gbtree"' model parameter in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

xgboost_gblinear: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.xgboost', 'booster = "gblinear"', 'resampling=rsmp("cv", folds = 5)', 'measure =

Table 3.3. Model parameters for voomStackPrio models.

Parameters	**voomStackPrio1**	**voomStackPrio2**
weights	sampleweights	sampleweights
family	binomial	binomial
type.measure	auc	auc
block1.penalization	TRUE	TRUE
lambda.type	lambda.min	lambda.1se
standardize	FALSE	FALSE
nfolds	5	5
cvoffset	TRUE	TRUE
cvoffsetnfolds	10	10

Table 3.4. Model parameters for voomStackIPF models.

Parameters	**voomStackIPF1**	**voomStackIPF2**	**voomStackIPF3**	**voomStackIPF4**	**voomStackIPF5**	**voomStackIPF6**	**voomStackIPF7**	**voomStackIPF8**	**voomStackIPF9**
weights	sampleweights	sampleweights	sampleweights	sampleweights	sampleweights	sampleweights	sampleweights	sampleweights	sampleweights
family	binomial	binomial	binomial	binomial	binomial	binomial	binomial	binomial	binomial
standardize	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
pf	c(1,1)	c(1,1)	c(1,1)	c(1,2)	c(1,2)	c(1,2)	c(2,1)	c(2,1)	c(2,1)
nfolds	5	5	5	5	5	5	5	5	5
ncv	10	10	10	10	10	10	10	10	10
type.measure	auc	auc	auc	auc	auc	auc	auc	auc	auc
alpha	0	0.5	1	0	0.5	1	0	0.5	1

msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the 'surv.xgboost' function and 'booster = "gblinear"' model parameter in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

xgboost_dart: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.xgboost', 'booster = "dart"', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the 'surv.xgboost' function and 'booster = "dart"' model parameter in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

elasticnet: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.glmnet', 'alpha = 0.5', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the 'surv.glmnet' function and 'alpha = 0.5' model parameter in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

lasso: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.glmnet', 'alpha = 1', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters

for tuning were set according to the parameters provided for the ‘surv.glmnet’ function and ‘alpha = 1’ model parameter in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

penalized: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.penalized', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.penalized’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

ridge: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.glmnet', 'alpha = 0', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.glmnet’ function and ‘alpha = 0’ model parameter in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

cforest: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.cforest', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.cforest’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

ctree: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For

feature selection, the parameters used were 'learner=surv.ctree', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the 'surv.ctree' function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

obliqueRSF: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.obliqueRSF', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the 'surv.obliqueRSF' function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

ranger: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.ranger', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the 'surv.ranger' function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

rfsrc: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.rfsrc', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters

for tuning were set according to the parameters provided for the ‘surv.rfsrc’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

rpart: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.rpart', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.rpart’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

svm: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. For feature selection, the parameters used were 'learner=surv.svm', 'resampling=rsmp("cv", folds = 5)', 'measure = msr("surv.cindex")', and 'evals20 = trm("evals", n_evals = 5)", and 'fselector = fs("random_search")' for internal feature selection in the **mlr3proba** package (215). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.svm’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

blackboost_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.blackboost’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

coxboost_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting

important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.coxboost’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

gbm_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.gbm’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

glmboost_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.glmboost’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

xgboost_gbtree_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.xgboost’ function and ‘booster = “gbtree”’ model parameter in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

xgboost_gblinear_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta**

package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.xgboost’ function and ‘booster = “gblinear”’ model parameter in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

xgboost_dart_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.xgboost’ function and ‘booster = “dart”’ model parameter in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

elasticnet_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.glmnet’ function and ‘alpha = 0.5’ model parameter in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

lasso_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.glmnet’ function and ‘alpha = 1’ model parameter in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

penalized_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta**

package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.penalized’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

ridge_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.glmnet’ function and ‘alpha = 0’ model parameter in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

cforest_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.cforest’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

ctree_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.ctree’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

obliqueRSF_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according

to the parameters provided for the ‘surv.obliqueRSF’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

ranger_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.ranger’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

rfsrc_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.rfsrc’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

rpart_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.rpart’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

svm_B: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, vst transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the ‘withTentative = TRUE’ parameter using the **Boruta** package (152). Model parameters and hyper-parameters for tuning were set according to the parameters provided for the ‘surv.svm’ function in Table 3.5. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **mlr3** package (211).

voomStackPrio1: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, voom transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the 'withTentative = TRUE' parameter using the **Boruta** package (152). After completing the pre-processing steps, the stacking algorithm was applied, and the dataset was transformed into a format suitable for input into the priority-Lasso algorithm. The model parameters defined for 'voomStackPrio1' are given in Table 3.3. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **prioritylasso** package (73).

voomStackPrio2: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, voom transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the 'withTentative = TRUE' parameter using the **Boruta** package (152). After completing the pre-processing steps, the stacking algorithm was applied, and the dataset was transformed into a format suitable for input into the priority-Lasso algorithm. The model parameters defined for 'voomStackPrio2' are given in Table 3.3. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **prioritylasso** package (73).

voomStackIPF1: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, voom transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the 'withTentative = TRUE' parameter using the **Boruta** package (152). After completing the pre-processing steps, the stacking algorithm was applied, and the dataset was transformed into a format suitable for input into the IPF-Lasso algorithm. The model parameters defined for 'voomStackIPF1' are given in Table 3.4. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **IPFlasso** package (74).

voomStackIPF2: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, voom transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the 'withTentative = TRUE' parameter using the **Boruta** package (152). After completing the pre-processing steps, the stacking algorithm was

applied, and the dataset was transformed into a format suitable for input into the IPF-Lasso algorithm. The model parameters defined for 'voomStackIPF2' are given in Table 3.4. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **IPFlasso** package (74).

voomStackIPF3: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, voom transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the 'withTentative = TRUE' parameter using the **Boruta** package (152). After completing the pre-processing steps, the stacking algorithm was applied, and the dataset was transformed into a format suitable for input into the IPF-Lasso algorithm. The model parameters defined for 'voomStackIPF3' are given in Table 3.4. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **IPFlasso** package (74).

voomStackIPF4: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, voom transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the 'withTentative = TRUE' parameter using the **Boruta** package (152). After completing the pre-processing steps, the stacking algorithm was applied, and the dataset was transformed into a format suitable for input into the IPF-Lasso algorithm. The model parameters defined for 'voomStackIPF4' are given in Table 3.4. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **IPFlasso** package (74).

voomStackIPF5: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, voom transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the 'withTentative = TRUE' parameter using the **Boruta** package (152). After completing the pre-processing steps, the stacking algorithm was applied, and the dataset was transformed into a format suitable for input into the IPF-Lasso algorithm. The model parameters defined for 'voomStackIPF5' are given in Table 3.4. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **IPFlasso** package (74).

voomStackIPF6: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, voom transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the 'withTentative = TRUE' parameter using the **Boruta** package (152). After completing the pre-processing steps, the stacking algorithm was applied, and the dataset was transformed into a format suitable for input into the IPF-Lasso algorithm. The model parameters defined for 'voomStackIPF6' are given in Table 3.4. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **IPFlasso** package (74).

voomStackIPF7: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, voom transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the 'withTentative = TRUE' parameter using the **Boruta** package (152). After completing the pre-processing steps, the stacking algorithm was applied, and the dataset was transformed into a format suitable for input into the IPF-Lasso algorithm. The model parameters defined for 'voomStackIPF7' are given in Table 3.4. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **IPFlasso** package (74).

voomStackIPF8: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, voom transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the 'withTentative = TRUE' parameter using the **Boruta** package (152). After completing the pre-processing steps, the stacking algorithm was applied, and the dataset was transformed into a format suitable for input into the IPF-Lasso algorithm. The model parameters defined for 'voomStackIPF8' are given in Table 3.4. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **IPFlasso** package (74).

voomStackIPF9: Filtering was performed for low-expressed genes after DESeq normalization. Subsequently, voom transformation and variance filtering were applied sequentially. Following that, Boruta feature selection was employed, selecting important features with the 'withTentative = TRUE' parameter using the **Boruta** package (152). After completing the pre-processing steps, the stacking algorithm was

applied, and the dataset was transformed into a format suitable for input into the IPF-Lasso algorithm. The model parameters defined for ‘voomStackIPF9’ are given in Table 3.4. 10 repeats of 5-fold cross-validation were performed to identify the optimal tuning parameter via the **IPFlasso** package (74).

Prediction and performance evaluation: The evaluation of survival algorithms involved the concordance index (Harrell's c-index) and integrated Brier score. For both metrics, the **survex** package (216) was utilized. The `c_index()` function was employed to calculate Harrell's concordance index, and the `integrated_brier_score()` function was used to assess the integrated Brier score metric.

Three distinct super lists have been generated for the Concordance Index, Integrated Brier Score, and the number of selected features by aggregating ordered lists based on their ranks. The **RankAggreg** package (217) was employed for this process, utilizing the `RankAggreg()` function with the Cross Entropy Monte Carlo method. Consequently, all survival algorithms, assessed against three different evaluation criteria, are ranked from best to worst performance. The consolidated version of these super lists is visually represented in a Venn diagram.

The performances of the models were also compared in terms of computation times.

3.2.4. Performance Evaluation Criteria

Sparsity, accuracy, and computational cost were the three parameters that were used to assess the performance of the model. In order to evaluate sparsity, one must determine how many features the model uses; models with fewer features are deemed to be more sparse. Model accuracy was evaluated using the metrics concordance index (Harrell's c-index) and integrated Brier score. Computational costs were computed, and models delivering results in the shortest time were highlighted.

A model in survival analysis predicts the risk of a specific event for each patient. The higher risk scores for patients with a shorter time-to-event determine the model's effectiveness. The concordance index (c-index) is a metric that measures the discriminating power of these risk models in survival analysis (218). When assessing this, it calculates the agreement between all pairs of samples. Two patients are

considered concordant if the predicted event risk by a model is lower for the patient who experiences the event later.

Let the risk scores of the patients be represented by φ. The concordance calculation for each pair of patients is conducted based on three scenarios. For patients *i* and *j*, the survival times are denoted as T_i and T_j, and the risk scores are φ_i and φ_j.

1. If both patients *i* and *j* are not censored,
   a. If $\varphi_i > \varphi_j$ and $T_i < T_j$, these patient pair is concordant and
   b. If $\varphi_i > \varphi_j$ and $T_i > T_j$, these patient pair is discordant.
2. If both patients *i* and *j* are censored, no calculation for this pair since it is unknown who first experienced the event.
3. If one of patients *i* and *j* is censored and patient *i* experience the event at time T_i and patient *j* is censored,
   a. If $T_i > T_j$, no calculation for this pair since it is unknown who first experienced the event.
   b. If $T_i < T_j$, since patient *i* experienced the event first
      i. If $\varphi_i > \varphi_j$, these patient pair is concordant and
      ii. If $\varphi_i < \varphi_j$, these patient pair is discordant.

$$Harrell's\ c-index = \frac{\#\ concordant\ pairs}{\#\ concordant\ pairs + \ \#\ discordant\ pairs}$$

The general formulation of Harrell's c-index is given below.

$$Harrell's\ c-index = \frac{\sum_{i,j} I\left(\tilde{T}_i > \tilde{T}_j\right).I\left(\varphi_j > \varphi_i\right).\Delta_j}{\sum_{i,j} I\left(\tilde{T}_i > \tilde{T}_j\right).\Delta_j}$$

Δ_j is a factor used to exclude non-comparable pairs of samples from the calculation, particularly when the shorter survival time is censored.

The concordance index is equivalent to the Area Under the Receiver Operating Characteristic Curve (AUC) in the presence of a binary outcome. This index ranges from zero to one. A c-index of 0.5 indicates that the risk model predicts randomly, and a c-index close to 1 indicates better discriminating power for the risk model.

The Brier score is used to assess the discrimination abilities of models and provide probabilistic results. It computes the mean squared error between the real classes and predicted risks for a dataset with binary outcomes. Subsequently, the Brier score was adapted for survival data (219). It is calculated as below.

$$Brier\ Score\ (t) = \frac{1}{n}\sum_{i=1}^{n} w_i(t)\,[\hat{y}_i(t) - y_i(t)]^2$$

$$w_i(t) = \begin{cases} \delta_i / C(y_i) & y_i \le t \\ 1 / C(y_i) & y_i > t \end{cases}$$

The probability of an event predicted for i^{th} sample is denoted as $\hat{y}_i(t)$, and the observed status outcome in the i^{th} sample is represented as $y_i(t)$ at time *t*. The calculation of $w_i(t)$ involves the use of the Kaplan-Meier estimator for the censoring distribution *C*. If the Brier score is close to 0, the predicted model is considered good. If it is around 0.25, the predicted model performs at random.

The Brier score assesses the accuracy of a survival function at a specific time. The integrated Brier score, obtained by integrating the Brier score across all follow-up times, is utilized, as a particular point of time can not be determined.

3.2.5. Computational Infrastructure

All analyses were conducted using the R programming language. We employed Version 2023.03.0+386 of the RStudio software for these analyses. Both R and RStudio are freely available as open-source software and can be installed on Windows, Macintosh, and Linux operating systems. To use RStudio, it is essential first to install R. With RStudio, you can easily execute R code, create graphical presentations, and access a history of your code. Details about the workstations used for running the analyses, including their respective features, can be found in Table 3.6.

Table 3.5. Characteristics of the compared survival models.

Group	Function	Package	Model-Parameters	Hyper-Parameters	Reference
Boosted survival model	surv.blackboost	mlr3 (211), mlr3proba (215), mlr3extralearners (220), mboost (221), pracma (222)	mstop = 100	family = "gehan", "cindex", mstop= 10 → 1000 nu= 0 → 0.1 mtry= 1 → max(feature_counts)	Bühlmann and Yu (2003) (223)
Random survival forest	surv.cforest	mlr3 (211), mlr3proba (215), mlr3extralearners (220), partykit (224), sandwich (225), coin (226)	ntree = 100	ntree= 250 → 2500, mtry= 1 → max(feature_counts)	Hothorn (2006) (194)
Boosted survival model	surv.coxboost	mlr3 (211), mlr3proba (215), mlr3extralearners (220), CoxBoost (227), pracma (222)		stepno= 500 → 1500	Binder (2009) (227)
Random survival forest	surv.ctree	mlr3 (211), mlr3proba (215), mlr3extralearners (220), partykit (224), coin (226), sandwich (225)		alpha= 0 → 1, abseps= 0 → 10, maxdepth= 1 → 16	Hothorn and Zeileis (2015) (224), Hothorn (2006) (194)
Boosted survival model	surv.gbm	mlr3 (211), mlr3proba (215), mlr3extralearners (220), gbm (228)	bag.fraction = 0.9	interaction.depth = 1 → 16	Friedman (2002) (229)
Boosted survival model	surv.glmboost	mlr3 (211), mlr3proba (215), mlr3extralearners (220), mboost (221), pracma (222)		family = "gehan", "cindex", mstop= 10 → 1000, nu= 0 → 0.1	Bühlmann and Yu (2003) (223)
Penalised Cox regression	surv.glmnet	mlr3 (211), mlr3proba (215), mlr3extralearners (220), glmnet (230)	alpha = 1, s = 0.01	lambda.min.ratio= 0 → 1	Friedman (2010) (231)
Penalised Cox regression	surv.glmnet	mlr3 (211), mlr3proba (215), mlr3extralearners (220), glmnet (230)	alpha = 0, s = 0.01	lambda.min.ratio= 0 → 1	Friedman (2010) (231)
Penalised Cox regression	surv.glmnet	mlr3 (211), mlr3proba (215), mlr3extralearners (220), glmnet (230)	alpha = 0.5, s = 0.01	lambda.min.ratio= 0 → 1	Friedman (2010) (231)

Random survival forest	surv.obliqueRSF	mlr3 (211), mlr3proba (215), mlr3extralearners (220), obliqueRSF (232), pracma (222)		alpha= 0 → 1, gamma= 0 → 1	Jaeger (2019) (233)
Penalised Cox regression	surv.penalized	mlr3 (211), mlr3proba (215), mlr3extralearners (220), penalized (234), pracma (222)	lambda1 = 10, lambda2 = 10	epsilon= 0 → 1	Goeman (2010) (234)
Random survival forest	surv.ranger	mlr3 (211), mlr3learners (235), ranger (236)		splitrule= "C", num.trees= 250 → 1000, mtry= 1 →max(feature_counts), min.node.size= 1 → 20	Breiman (2001) (38)
Random survival forest	surv.rfsrc	mlr3 (211), mlr3proba (215), mlr3extralearners (220), randomForestSRC (237), pracma (222)		ntree= 250 → 2500, mtry= 1 → max(feature_counts), nodesize= 1 → 20	Ishwaran (2008) (39), Breiman (2001) (38)
Random survival forest	surv.rpart	mlr3 (211), mlr3proba (215), rpart (238), distr6 (239), survival (240)		minbucket= 1 → 20, maxdepth= 2 → 30	Breiman (1984) (174)
Support vector machine	surv.svm	mlr3 (211), mlr3proba (215), mlr3extralearners (220), survivalsvm (241)	type = "hybrid", diff.meth = "makediff3", kernel = "lin_kernel", gamma.mu = c(100,1000)	sigf= 2 → 12, maxiter= 20 → 50, margin= 0.01 → 0.1, bound= 5 → 15	Van Belle (2011) (167)
Boosted survival model	surv.xgboost	mlr3 (211), mlr3learners (235), xgboost (242)	booster = "gbtree"	alpha= 0 → 1, eta= 0 → 1, gamma=0 → 1, lambda=0 → 2, nrounds= 1 → 16	Chen (2016) (198)
Boosted survival model	surv.xgboost	mlr3 (211), mlr3learners (235), xgboost (242)	booster = "gblinear"	alpha= 0 → 1, eta= 0 → 1, lambda=0 → 2, nrounds= 1 → 16	Chen (2016) (198)
Boosted survival model	surv.xgboost	mlr3 (211), mlr3learners (235), xgboost (242)	booster = "dart"	alpha= 0 → 1, eta= 0 → 1, gamma=0 → 1, lambda=0 → 2, nrounds= 1 → 16	Chen (2016) (198)

Table 3.6. Characteristics of the workstations employed for analysis.

Workstation	Operating System	CPU	GPU	Memory	Number of Cores
Erciyes University, Department of Biostatistics	Windows 10	Intel i7-4790 3.60GHz	Intel HD Graphics 4600	16 GB	4 cores 8 logical processors
Erciyes University, Ziya Eren Drug Research and Application Center (ERFARMA)	Ubuntu 20.04 - Linux	AMD EPYC 7742 (x2) – 256 CPU	2xTesla V100S 32GB	2 TB	256
Erciyes University, Department of Information Technology	Windows 10	Intel(R) Xeon(R) 32 CPU E5-2650 V4 @ 2.20 GHz	-	350 GB	30 cores
Personal Computer	Windows 10	Intel(R) Core™ i5-8265U CPU, 1.60GHz, 1800 Mhz	-	8 GB	4

3.3. MLSeqSurv R Package

The voomStackPrio and voomStackIPF algorithms have a R package called **MLSeqSurv**. With the help of this package, researchers can do survival analyses on RNA-seq data by incorporating both newly created and previously published survival algorithms. Researcher input datasets (training and test datasets) are required in order to use the **MLSeqSurv** R package. These datasets can be submitted in formats such as .csv, .xlsx, and .txt. Once users input the datasets and chosen survival algorithm and its parameters, the package automatically trains the model tailored to the training set. After model training, the package calculates survival probabilities for the test data at specified time points. Additionally, MLSeqSurv provides users with individual survival curves for the test data. The source code for this package is available on the official website at https://github.com/gokmenzararsiz/MLSeqSurv. Following the transfer of the **MLSeqSurv** package to the R BIOCONDUCTOR repository, installation can be achieved using the following code.

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("MLSeqSurv")
```

The MLSeqSurv R packages used are **mlr3** (211), **mlr3proba** (215), **mlr3learners** (235), **mlr3extralearners** (220), **mlr3verse** (243), **mlr3tuningspaces** (244), **mlr3fselect** (214), **limma** (150), **edgeR** (97), **DESeq2** (68), **survival** (240), **prioritylasso** (73), **ipflasso** (74), **mboost** (221), **pracma** (222), **partykit** (224), **sandwich** (225), **coin** (226), **gbm** (228), **glmnet** (230), **CoxBoost** (227), **obliqueRSF** (232), **penalized** (234), **ranger** (236), **rpart** (238), **distr6** (239), **randomForestSRC** (237), **survivalsvm** (241), **xgboost** (242), **survex** (216), **Boruta** (152).

4. RESULTS

4.1. Concordance Index, Integrated Brier Score, and Selected Features for Real RNA-Seq Datasets

The results of the 12 real RNA-seq survival cancer datasets, as outlined in Table 3.9, are depicted graphically in Figures 4.1 through 4.12 and elaborated upon in tables ranging from Table 4.1 to Table 4.12. The concordance index, integrated Brier score, and the number of selected features for each dataset are illustrated in boxplots. The methods compared in these graphs are classified into five main groups: Boosted survival models (Boosted), Penalized Cox regression models (Penalized), Random survival forests (RSF), Survival support vector machine (SVM), and voom-based stacking lasso methods (voomStackLasso).

Boosted survival models (Boosted) consist of algorithms such as blackboost, coxboost, gamboost, gbm, glmboost, xgboost (including dart, gblinear, and gbtree), and are represented in light pink. Penalized Cox regression models (Penalized) include elasticnet, lasso, penalized, ridge algorithms, depicted in dark khaki. Random survival forests (RSF) comprise cforest, ctree, obliqueRSF, ranger, rfsrc, rpart algorithms, shown in green. Survival support vector machine (SVM) is represented in blue. Results from the existing survival algorithms in the literature include outcomes from both the internal feature selection algorithm, individually applied for each algorithm in the **mlr3proba** package (215), and the feature selection process in the **Boruta** package (152).

voom-based stacking lasso models (voomStackLasso), developed within the scope of this study, include voomstackPrio1, voomstackPrio2, voomstackIPF1, voomstackIPF2, voomstackIPF3, voomstackIPF4, voomstackIPF5, voomstackIPF6, voomstackIPF7, voomstackIPF8, and voomstackIPF9, depicted in purple.

Summary statistics for the concordance index, integrated Brier score, and the number of selected features are provided in the tables. The survival algorithms, whose performance is compared, are listed in the table rows. The columns present the mean, standard deviation, median, 1st-3rd quartile, minimum, and maximum statistics for the concordance index, integrated Brier score, and the number of selected features. The tables are formatted with bold to draw attention to the highest values. The midpoints

of the lines in the boxplots stand for the median, the bottom point for the lowest value, and the top point for the maximum value.

The concordance index, integrated Brier score, and the number of selected features for Adrenocortical Carcinoma (ACC) data are depicted in Figure 4.1, with related summary statistics presented in Table 4.1. Upon examination of the graph and table, it was observed that the cforest algorithm, when applied to internal feature selection, exhibited the highest mean concordance index for ACC data at 0.866. Among the methods applied to internal feature selection, the highest mean concordance index values were observed for cforest (0.866±0.044), blackboost (0.861±0.050), ridge (0.860±0.042), rfsrc (0.857±0.052), svm (0.857±0.046), and ranger (0.854±0.073) algorithms. Conversely, the lowest mean concordance index values were attributed to ctree (0.742±0.090) and rpart (0.758±0.089) algorithms. Among the methods from the literature employing Boruta feature selection, the cforest (cforest_B) (0.854±0.062), ridge (ridge_B) (0.856±0.057) and xgboost (with booster= "gblinear") (xgboost_gblinear_B) (0.852±0.047) algorithms demonstrated the highest mean concordance index. On the other hand, the svm (svm_B) algorithm (0.555±0.245) exhibited the lowest mean concordance index. Among the voomStackLasso methods, the voomStackIPF1 (0.855±0.054), voomStackIPF4 (0.854±0.053), and voomStackIPF7 (0.854±0.053) algorithms showed the highest mean concordance index values, while the voomStackPrio2 algorithm (0.737±0.075) displayed the lowest mean concordance index.

It was observed that the penalized algorithm, when utilized with Boruta feature selection, resulted in the lowest mean integrated Brier score for ACC data, recorded at 0.131. Within the category of methods applied to internal feature selection, the penalized (0.142±0.048) and cforest (0.158±0.027) algorithms demonstrated the lowest mean integrated Brier scores, while gbm (0.375±0.079), lasso (0.366±0.181), blackboost (0.359±0.064), and svm (0.339±0.069) algorithms displayed the highest mean integrated Brier scores. In the group of methods from the literature employing Boruta feature selection, the penalized (penalized_B) (0.131±0.032), ranger (ranger_B) (0.140±0.031), and cforest (cforest_B) (0.142±0.033) algorithms showcased the lowest mean integrated Brier scores, while lasso (0.479±0.150) and elasticnet (0.452±0.159) algorithms presented the highest mean integrated Brier scores. Among the voomStackLasso methods, voomStackIPF1 (0.134±0.027),

voomStackIPF7 (0.134±0.026), and voomStackIPF4 (0.135±0.028) algorithms demonstrated the lowest mean integrated Brier scores, whereas the voomStackPrio1 algorithm (0.170±0.071) displayed the highest mean integrated Brier score.

voomStackLasso algorithms showed the lowest mean number of selected features for ACC data (52.50±7.83). These were closely followed by the methods in the literature that utilized Boruta feature selection (54.07±7.62). Regarding internal feature selection methods, the algorithm with the lowest mean number of features was rpart (600.77±427.46), while the algorithm with the highest mean number of features was obliqueRSF (1137.10±581.78).

The concordance index, Integrated Brier Score, and the number of selected features for Cervical Squamous Cell Carcinoma and Endocervical Adenocarcinoma (CESC) data are depicted in Figure 4.2, with related summary statistics presented in Table 4.2. Upon examination of both the graph and table, it was observed that the ridge algorithm, when applied to internal feature selection, exhibited the highest mean concordance index for CESC data at 0.686. Within the category of methods applied to internal feature selection, the highest mean concordance index values were observed for ridge (0.686±0.052), penalized (0.667±0.056), and cforest (0.662±0.054). Conversely, the lowest mean concordance indices were attributed to ctree (0.557±0.053) and rpart (0.573±0.073) algorithms. Among the methods from the literature employing Boruta feature selection, the ranger (ranger_B) (0.643±0.066) and ridge (ridge_B) (0.632±0.062) algorithms demonstrated the highest mean concordance index, while the rpart (rpart_B) (0.546±0.082) and ctree (ctree_B) (0.547±0.078) algorithms exhibited the lowest mean concordance index values. Among the voomStackLasso methods, the voomStackIPF1 (0.660±0.047) and voomStackIPF7 (0.659±0.047) algorithms showed the highest mean concordance index values, while the voomStackPrio2 algorithm (0.628±0.055) displayed the lowest mean concordance index.

Upon reviewing the integrated Brier score results for CESC data, it was evident that methods from the literature, where both internal feature selection and Boruta feature selection were applied, consistently yielded high results. The voomStackLasso methods yield the lowest integrated Brier score results. It was noted that among these algorithms, the voomStackIPF4 exhibited the lowest mean integrated Brier score for CESC data, at 0.191. This was followed by voomStackIPF1

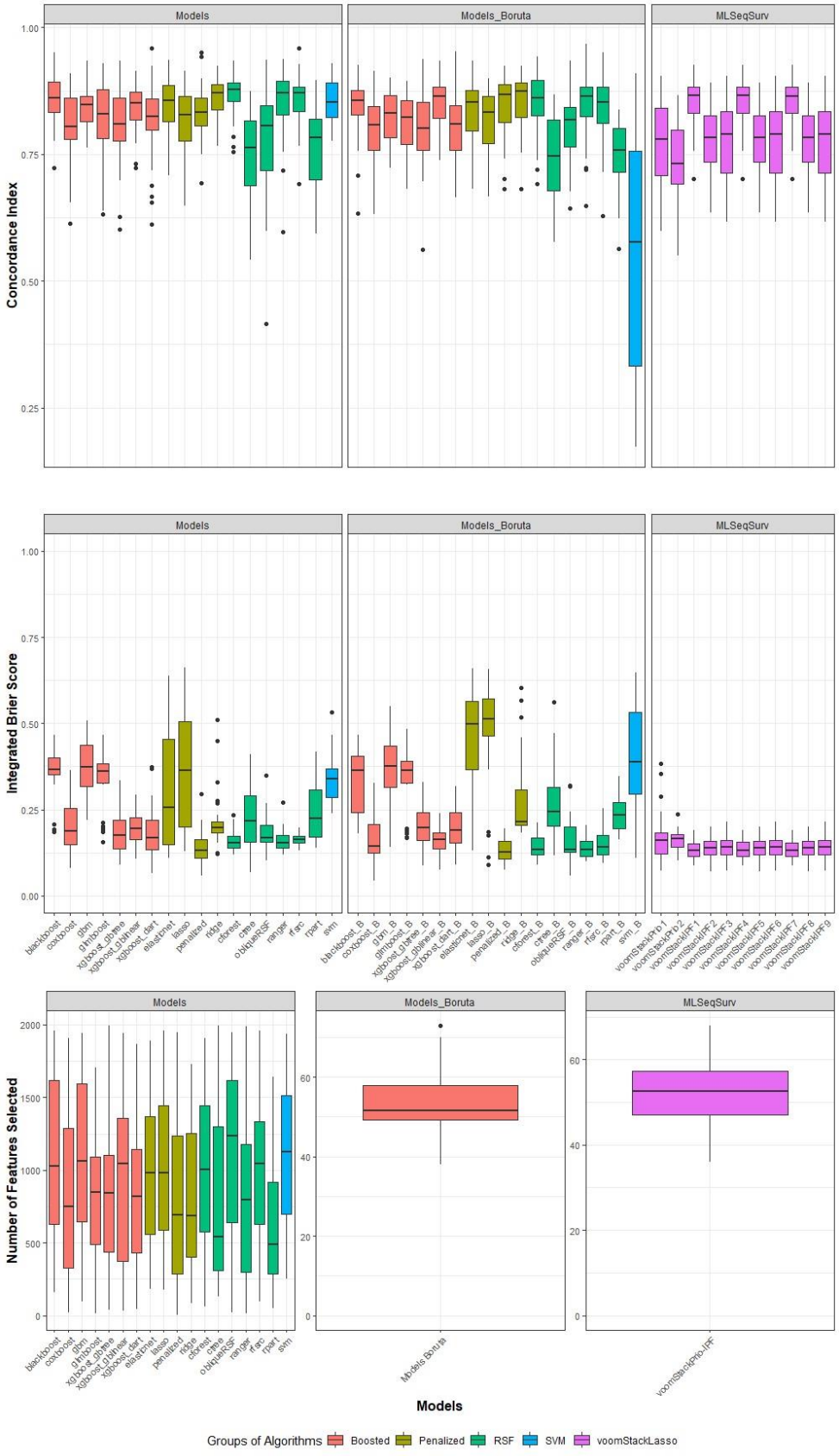


Figure 4.1. The concordance index, integrated Brier score, and the number of selected features for ACC.

Table 4.1. The summary statistics of concordance index, integrated Brier score and the number of features selected for ACC.

Groups of Algorithms	Models	Concordance Index			Integrated Brier Score			The Number of Features Selected		
		Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max
Models	**Blackboost**	**0.861±0.050**	0.861 (0.832-0.892)	0.723-0.950	0.359±0.064	0.366 (0.349-0.402)	0.186-0.468	1081.83±570.29	1027.00 (597.75-1644.00)	157-1960
	Cforest	**0.866±0.044**	0.878 (0.851-0.892)	0.755-0.933	**0.158±0.027**	0.154 (0.138-0.174)	0.119-0.233	1025.27±570.19	1006.50 (541.25-1492.00)	62-1907
	Coxboost	0.803±0.075	0.805 (0.773-0.866)	0.614-0.908	0.209±0.081	0.187 (0.144-0.267)	0.079-0.364	867.10±574.20	748.00 (270.00-1332.50)	23-1908
	Ctree	0.742±0.090	0.764 (0.686-0.824)	0.543-0.875	0.225±0.090	0.218 (0.156-0.299)	0.067-0.411	806.90±602.60	544.00 (309.75-1359.50)	131-1995
	Elasticnet	0.844±0.058	0.856 (0.812-0.887)	0.708-0.936	0.312±0.176	0.255 (0.149-0.473)	0.111-0.639	992.90±522.90	979.00 (540.00-1426.00)	183-1891
	Gbm	0.841±0.043	0.848 (0.813-0.865)	0.763-0.933	0.375±0.079	0.373 (0.316-0.441)	0.219-0.508	1089.47±578.86	1063.50 (627.00-1623.25)	93-1943
	Glmboost	0.817±0.076	0.830 (0.775-0.880)	0.632-0.930	0.335±0.086	0.363 (0.295-0.390)	0.157-0.468	850.87±481.50	850.50 (454.25-1139.25)	15-1706
	Lasso	0.818±0.059	0.828 (0.768-0.866)	0.649-0.914	0.366±0.181	0.364 (0.192-0.508)	0.129-0.662	1021.77±537.09	982.00 (555.00-1480.75)	175-1960
	ObliqueRSF	0.780±0.107	0.806 (0.717-0.849)	0.416-0.936	0.184±0.051	0.168 (0.156-0.209)	0.101-0.349	1137.10±581.78	1238.00 (600.25-1631.50)	22-1948
	Penalized	0.829±0.058	0.832 (0.798-0.862)	0.692-0.950	**0.142±0.048**	0.131 (0.110-0.170)	0.057-0.295	776.67±551.31	691.50 (284.75-1274.25)	5-1946
	Ranger	**0.854±0.073**	0.870 (0.822-0.899)	0.597-0.938	0.160±0.030	0.155 (0.138-0.177)	0.119-0.271	815.33±575.74	798.00 (284.25-1278.00)	14-1988
	Rfsrc	**0.857±0.052**	0.870 (0.831-0.884)	0.691-0.958	0.163±0.016	0.164 (0.152-0.174)	0.131-0.198	1050.30±493.23	1043.00 (614.00-1344.75)	97-1959
	Ridge	**0.860±0.042**	0.871 (0.833-0.888)	0.766-0.924	0.218±0.083	0.198 (0.179-0.220)	0.122-0.511	807.27±515.48	683.50 (394.00-1265.00)	84-1728
	Rpart	0.758±0.089	0.783 (0.691-0.821)	0.593-0.896	0.241±0.078	0.225 (0.169-0.307)	0.138-0.418	600.77±427.46	489.50 (260.25-928.25)	51-1638
	Svm	**0.857±0.046**	0.853 (0.818-0.893)	0.776-0.930	0.339±0.069	0.339 (0.280-0.374)	0.239-0.532	1116.87±493.32	1124.00 (682.75-1539.00)	250-1935
	Xgboost (dart)	0.810±0.082	0.825 (0.783-0.865)	0.611-0.958	0.183±0.074	0.168 (0.132-0.221)	0.065-0.374	817.97±498.17	821.50 (382.25-1195.25)	43-1869
	Xgboost (gblinear)	0.842±0.048	0.852 (0.816-0.874)	0.724-0.914	0.194±0.050	0.195 (0.161-0.228)	0.108-0.292	993.07±599.29	1046.00 (362.75-1395.75)	33-1939
	Xgboost (gbtree)	0.806±0.075	0.809 (0.770-0.864)	0.601-0.934	0.182±0.056	0.175 (0.131-0.220)	0.090-0.334	835.63±552.05	844.00 (413.50-1155.25)	39-1994
Models Boruta	**Blackboost**	0.841±0.062	0.857 (0.825-0.878)	0.633-0.925	0.333±0.089	0.364 (0.214-0.405)	0.180-0.468	**54.07±7.62**	51.50 (49.00-58.50)	38-73
	Cforest	**0.854±0.062**	0.861 (0.824-0.899)	0.691-0.942	**0.142±0.033**	0.134 (0.118-0.170)	0.090-0.211			
	Coxboost	0.800±0.067	0.808 (0.752-0.851)	0.632-0.914	0.167±0.073	0.145 (0.118-0.212)	0.043-0.328			
	Ctree	0.739±0.086	0.746 (0.674-0.819)	0.576-0.868	0.269±0.101	0.244 (0.200-0.322)	0.117-0.561			
	Elasticnet	0.831±0.060	0.852 (0.793-0.877)	0.681-0.933	0.452±0.159	0.498 (0.345-0.568)	0.133-0.659			
	Gbm	0.824±0.049	0.831 (0.781-0.866)	0.723-0.901	0.368±0.102	0.375 (0.314-0.438)	0.141-0.550			
	Glmboost	0.809±0.062	0.822 (0.765-0.860)	0.681-0.894	0.340±0.089	0.364 (0.322-0.396)	0.169-0.484			
	Lasso	0.813±0.065	0.833 (0.761-0.867)	0.667-0.898	0.479±0.150	0.513 (0.461-0.573)	0.091-0.656			
	ObliqueRSF	0.802±0.066	0.817 (0.758-0.844)	0.644-0.934	0.157±0.063	0.134 (0.122-0.203)	0.059-0.320			
	Penalized	0.845±0.062	0.867 (0.810-0.888)	0.681-0.924	**0.131±0.032**	0.127 (0.105-0.162)	0.074-0.195			
	Ranger	0.844±0.067	0.864 (0.822-0.883)	0.649-0.967	**0.140±0.031**	0.133 (0.113-0.161)	0.099-0.204			
	Rfsrc	0.842±0.067	0.852 (0.809-0.883)	0.628-0.950	0.147±0.035	0.141 (0.117-0.176)	0.095-0.253			
	Ridge	**0.856±0.057**	0.874 (0.821-0.893)	0.681-0.924	0.281±0.124	0.214 (0.202-0.328)	0.182-0.603			
	Rpart	0.747±0.069	0.758 (0.705-0.802)	0.564-0.838	0.239±0.050	0.234 (0.193-0.277)	0.164-0.346			
	Svm	0.555±0.245	0.577 (0.314-0.767)	0.174-0.908	0.407±0.149	0.389 (0.286-0.536)	0.110-0.648			
	Xgboost (dart)	0.800±0.066	0.809 (0.751-0.847)	0.665-0.953	0.194±0.061	0.190 (0.147-0.241)	0.091-0.318			
	Xgboost (gblinear)	**0.852±0.047**	0.865 (0.817-0.886)	0.738-0.933	0.164±0.037	0.163 (0.135-0.186)	0.074-0.240			
	Xgboost (gbtree)	0.800±0.073	0.802 (0.756-0.856)	0.563-0.938	0.199±0.062	0.198 (0.160-0.250)	0.087-0.330			
MLSeqSurv	**voomStackPrio1**	0.771±0.082	0.779 (0.705-0.842)	0.598-0.904	0.170±0.071	0.161 (0.122-0.186)	0.074-0.382	**52.50±7.83**	52.50 (46.75-58.25)	36-68
	voomStackPrio2	0.737±0.075	0.731 (0.688-0.801)	0.551-0.867	0.162±0.028	0.165 (0.140-0.179)	0.103-0.237			
	voomStackIPF1	**0.855±0.054**	0.866 (0.823-0.888)	0.702-0.925	**0.134±0.027**	0.132 (0.113-0.155)	0.087-0.190			
	voomStackIPF2	0.776±0.070	0.783 (0.727-0.830)	0.636-0.891	**0.139±0.029**	0.139 (0.118-0.158)	0.070-0.200			
	voomStackIPF3	0.775±0.081	0.789 (0.711-0.837)	0.617-0.904	**0.142±0.031**	0.142 (0.118-0.163)	0.072-0.214			
	voomStackIPF4	**0.854±0.053**	0.866 (0.821-0.888)	0.702-0.925	**0.135±0.028**	0.132 (0.113-0.157)	0.087-0.190			
	voomStackIPF5	0.776±0.070	0.783 (0.727-0.830)	0.636-0.891	**0.139±0.029**	0.139 (0.118-0.158)	0.070-0.200			
	voomStackIPF6	0.774±0.081	0.789 (0.711-0.837)	0.617-0.904	**0.142±0.031**	0.142 (0.118-0.163)	0.072-0.214			
	voomStackIPF7	**0.854±0.053**	0.863 (0.823-0.888)	0.702-0.925	**0.134±0.026**	0.132 (0.113-0.155)	0.087-0.190			
	voomStackIPF8	0.776±0.070	0.783 (0.727-0.830)	0.636-0.891	**0.139±0.029**	0.139 (0.118-0.158)	0.070-0.200			
	voomStackIPF9	0.775±0.081	0.789 (0.711-0.837)	0.617-0.904	**0.142±0.031**	0.142 (0.118-0.163)	0.072-0.214			

(0.192±0.016), voomStackIPF7 (0.192±0.016), voomStackPrio2 (0.195±0.017), and voomStackIPF5 (0.195±0.020). The voomStackPrio1 algorithm (0.224±0.053) displayed the highest mean integrated Brier score. Within the category of methods applied to internal feature selection, the penalized algorithm (0.200±0.023) demonstrated the lowest mean integrated Brier score, whereas gbm (0.390±0.108), svm (0.371±0.074), and blackboost (0.366±0.072) algorithms displayed the highest mean integrated Brier scores. In the group of methods from the literature employing Boruta feature selection, the ridge (ridge_B) (0.212±0.008), penalized (penalized_B) (0.214±0.024) and ranger (ranger_B) (0.214±0.016) algorithms showcased the lowest mean integrated Brier scores, while svm (svm_B) (0.456±0.105), gbm (gbm_B) (0.411±0.120) and blackboost (blackboost_B) (0.365±0.071) algorithms presented the highest mean integrated Brier scores.

Among the voomStackLasso algorithms, the mean number of selected features for CESC data was the lowest (11.70±4.72). These were closely followed by the methods in the literature that utilized Boruta feature selection (12.63±4.72). In terms of internal feature selection methods, the algorithm with the lowest mean number of features was elasticnet (737.30±453.70), while the algorithm with the highest mean number of features was xgboost (with booster= “gblinear”) (1335.53±491.05).

The concordance index, integrated Brier score, and the number of selected features for Esophageal Carcinoma (ESCA) data are depicted in Figure 4.3, with related summary statistics presented in Table 4.3. After examining both the graph and the table for ESCA data, it was observed that the ranger algorithm, when applied to internal feature selection, achieved the highest mean concordance index at 0.580. The second-highest value was recorded by the voomStackIPF7 algorithm, with an average of 0.560. Within the category of methods applied to internal feature selection, the highest mean concordance indices were observed for ranger (0.580±0.060) and svm (0.559±0.057). Conversely, the lowest mean concordance index values were attributed to penalized (0.478±0.064) and blackboost (0.493±0.075) algorithms. In the group of methods from the literature employing Boruta feature selection, the rpart (rpart_B) (0.519±0.076) and obliqueRSF (obliqueRSF_B) (0.517±0.078) algorithms demonstrated the highest mean concordance index values, while the xgboost (with booster= “gblinear”) (xgboost_gblinear_B) (0.476±0.053) and svm (svm_B)

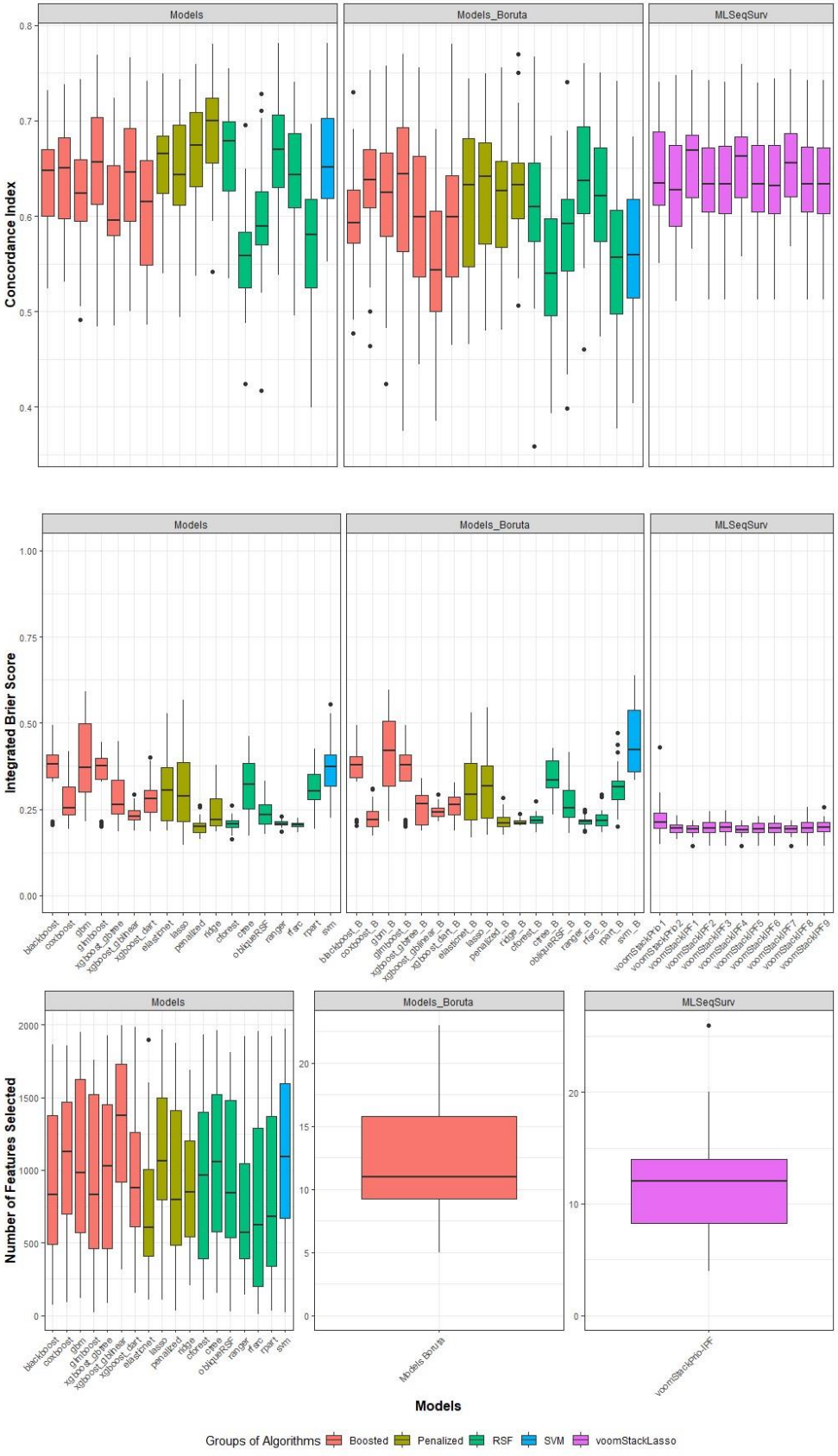


Figure 4.2. The concordance index, integrated Brier score, and the number of selected features for CESC.

Table 4.2. The summary statistics of concordance index, integrated Brier score and the number of features selected for CESC.

Groups of Algorithms	Models	Concordance Index			Integrated Brier Score			The Number of Features Selected		
		Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max
Models	**Blackboost**	0.636±0.050	0.647 (0.599-0.672)	0.524-0.732	0.366±0.072	0.381 (0.340-0.411)	0.204-0.494	914.17±530.80	832.50 (472.25-1420.75)	71-1863
	Cforest	**0.662±0.054**	0.679 (0.623-0.700)	0.535-0.755	0.207±0.019	0.207 (0.197-0.217)	0.164-0.261	935.27±565.19	963.00 (345.50-1410.75)	108-1929
	Coxboost	0.639±0.054	0.651 (0.595-0.684)	0.532-0.738	0.271±0.054	0.254 (0.232-0.316)	0.193-0.417	1038.60±547.91	1128.00 (688.75-1481.75)	88-1858
	Ctree	0.557±0.053	0.559 (0.522-0.584)	0.424-0.696	0.319±0.087	0.323 (0.246-0.386)	0.172-0.463	1053.77±548.88	1055.00 (553.00-1528.75)	152-1959
	Elasticnet	0.651±0.054	0.665 (0.621-0.687)	0.540-0.749	0.304±0.090	0.306 (0.218-0.374)	0.188-0.528	737.30±453.70	603.00 (384.25-1018.75)	105-1896
	Gbm	0.623±0.066	0.624 (0.589-0.665)	0.492-0.743	0.390±0.108	0.370 (0.297-0.499)	0.215-0.592	1066.67±609.77	984.00 (566.50-1635.25)	118-1950
	Glmboost	0.655±0.067	0.657 (0.608-0.706)	0.485-0.768	0.348±0.077	0.376 (0.334-0.401)	0.200-0.443	917.50±599.38	831.00 (428.50-1531.25)	18-1756
	Lasso	0.645±0.061	0.643 (0.609-0.697)	0.494-0.743	0.306±0.102	0.287 (0.213-0.390)	0.147-0.567	1128.70±520.11	1060.50 (786.00-1583.25)	106-1969
	ObliqueRSF	0.597±0.060	0.589 (0.569-0.627)	0.417-0.728	0.241±0.042	0.234 (0.207-0.266)	0.179-0.332	948.03±533.00	841.50 (525.50-1523.00)	24-1812
	Penalized	**0.667±0.056**	0.675 (0.629-0.710)	0.538-0.759	**0.200±0.023**	0.199 (0.181-0.211)	0.164-0.261	883.80±581.86	797.00 (470.25-1487.50)	32-1871
	Ranger	0.659±0.056	0.670 (0.627-0.707)	0.538-0.781	0.209±0.009	0.208 (0.204-0.215)	0.186-0.230	768.13±520.49	570.50 (381.25-1094.00)	141-1918
	Rfsrc	0.642±0.056	0.643 (0.606-0.688)	0.496-0.741	0.206±0.009	0.206 (0.199-0.211)	0.183-0.224	758.63±624.16	624.00 (188.50-1322.25)	8-1957
	Ridge	**0.686±0.052**	0.700 (0.655-0.725)	0.542-0.780	0.241±0.049	0.220 (0.202-0.281)	0.186-0.379	869.97±415.53	851.00 (538.75-1224.25)	203-1691
	Rpart	0.573±0.073	0.581 (0.517-0.620)	0.399-0.696	0.308±0.057	0.302 (0.278-0.355)	0.193-0.424	845.83±584.01	683.00 (327.00-1402.75)	32-1922
	Svm	0.658±0.058	0.652 (0.618-0.706)	0.553-0.781	0.371±0.074	0.374 (0.318-0.410)	0.224-0.556	1123.77±540.47	1091.00 (655.50-1602.75)	19-1972
	Xgboost (dart)	0.607±0.075	0.615 (0.544-0.665)	0.486-0.741	0.275±0.054	0.280 (0.238-0.307)	0.185-0.401	955.90±467.37	877.50 (594.00-1304.75)	156-1984
	Xgboost (gblinear)	0.630±0.081	0.646 (0.591-0.693)	0.500-0.766	0.234±0.022	0.230 (0.219-0.249)	0.189-0.294	1335.53±491.05	1376.50 (837.50-1754.50)	318-1997
	Xgboost (gbtree)	0.607±0.059	0.596 (0.577-0.657)	0.485-0.723	0.285±0.070	0.264 (0.235-0.341)	0.184-0.447	978.17±566.64	1027.00 (437.00-1475.75)	82-1926
Models Boruta	**Blackboost**	0.592±0.063	0.593 (0.569-0.631)	0.478-0.730	0.365±0.071	0.379 (0.340-0.405)	0.203-0.494	**12.63±4.72**	11.00 (9.00-16.00)	5-23
	Cforest	0.614±0.080	0.610 (0.571-0.658)	0.359-0.767	0.218±0.019	0.218 (0.209-0.230)	0.182-0274			
	Coxboost	0.629±0.069	0.638 (0.605-0.670)	0.464-0.753	0.227±0.038	0.219 (0.200-0.248)	0.173-0.309			
	Ctree	0.547±0.078	0.540 (0.493-0.600)	0.394-0.684	0.341±0.056	0.334 (0.311-0.394)	0.233-0.428			
	Elasticnet	0.617±0.082	0.633 (0.544-0.686)	0.466-0.744	0.309±0.101	0.292 (0.216-0.395)	0.168-0.530			
	Gbm	0.621±0.076	0.625 (0.577-0.699)	0.424-0.757	0.411±0.120	0.420 (0.301-0.518)	0.215-0.596			
	Glmboost	0.623±0.093	0.644 (0.557-0.694)	0.375-0.769	0.350±0.086	0.379 (0.303-0.411)	0.200-0.494			
	Lasso	0.623±0.076	0.642 (0.561-0.679)	0.480-0.749	0.324±0.109	0.318 (0.222-0.384)	0.175-0.545			
	ObliqueRSF	0.579±0.075	0.592 (0.541-0.620)	0.399-0.740	0.268±0.061	0.253 (0.226-0.310)	0.180-0.416			
	Penalized	0.621±0.067	0.627 (0.565-0.669)	0.481-0.756	**0.214±0.024**	0.210 (0.198-0.229)	0.176-0.283			
	Ranger	**0.643±0.066**	0.637 (0.600-0.699)	0.460-0.760	**0.214±0.016**	0.214 (0.207-0.221)	0.186-0.250			
	Rfsrc	0.615±0.070	0.621 (0.571-0.672)	0.474-0.750	0.221±0.026	0.218 (0.202-0.236)	0.184-0.294			
	Ridge	**0.632±0.062**	0.632 (0.594-0.660)	0.507-0.769	**0.212±0.008**	0.210 (0.206-0.217)	0.203-0.237			
	Rpart	0.546±0.082	0.557 (0.492-0.614)	0.378-0.741	0.317±0.060	0.315 (0.277-0.336)	0.200-0.471			
	Svm	0.563±0.078	0.560 (0.514-0.622)	0.404-0.683	0.456±0.105	0.424 (0.356-0.543)	0.335-0.638			
	Xgboost (dart)	0.599±0.078	0.599 (0.531-0.648)	0.465-0.780	0.263±0.039	0.262 (0.235-0.287)	0.188-0.326			
	Xgboost (gblinear)	0.558±0.069	0.543 (0.500-0.608)	0.386-0.691	0.244±0.019	0.241 (0.230-0.254)	0.214-0.294			
	Xgboost (gbtree)	0.601±0.077	0.599 (0.533-0.670)	0.445-0.756	0.254±0.050	0.265 (0.202-0.292)	0.187-0.338			
MLSeqSurv	**voomStackPrio1**	0.645±0.051	0.634 (0.609-0.694)	0.551-0.741	0.224±0.053	0.211 (0.193-0.242)	0.148-0.431	**11.70±4.72**	12.00 (8.00-14.00)	4-26
	voomStackPrio2	0.628±0.055	0.627 (0.589-0.675)	0.511-0.748	**0.195±0.017**	0.195 (0.183-0.206)	0.162-0.232			
	voomStackIPF1	**0.660±0.047**	0.668 (0.619-0.685)	0.566-0.753	**0.192±0.016**	0.192 (0.184-0.204)	0.145-0.218			
	voomStackIPF2	0.639±0.050	0.633 (0.603-0.678)	0.513-0.742	0.197±0.022	0.196 (0.181-0.213)	0.144-0.245			
	voomStackIPF3	0.638±0.050	0.633 (0.602-0.678)	0.513-0.741	0.198±0.022	0.197 (0.183-0.212)	0.145-0.247			
	voomStackIPF4	0.658±0.049	0.663 (0.616-0.685)	0.558-0.759	**0.191±0.016**	0.190 (0.183-0.204)	0.145-0.217			
	voomStackIPF5	0.640±0.049	0.633 (0.603-0.677)	0.513-0.740	**0.195±0.020**	0.193 (0.181-0.209)	0.144-0.229			
	voomStackIPF6	0.639±0.049	0.632 (0.602-0.677)	0.513-0.744	0.196±0.020	0.195 (0.181-0.210)	0.145-0.231			
	voomStackIPF7	**0.659±0.047**	0.656 (0.619-0.687)	0.568-0.754	**0.192±0.016**	0.193 (0.184-0.204)	0.145-0.218			
	voomStackIPF8	0.638±0.050	0.633 (0.603-0.678)	0.513-0.742	0.198±0.023	0.196 (0.181-0.214)	0.144-0.256			
	voomStackIPF9	0.637±0.050	0.633 (0.602-0.678)	0.513-0.742	0.199±0.023	0.197 (0.183-0.213)	0.145-0.257			

(0.481±0.058) algorithms exhibited the lowest mean concordance index values. Among the voomStackLasso methods, the voomStackIPF7 (0.560±0.063) and voomStackIPF1 (0.558±0.064) algorithms showed the highest mean concordance index values, while the voomStackPrio2 algorithm (0.530±0.122) displayed the lowest mean concordance index.

It was observed that the cforest algorithm, when applied to internal feature selection, achieved the lowest mean integrated Brier score at 0.182 for ESCA data. Following this, the penalized (penalized_B) and ridge (ridge_B) algorithms, both with Boruta feature selection applied, yielded an integrated Brier score of 0.183. Within the category of methods applied to internal feature selection, the cforest (0.182±0.025) and penalized (0.193±0.025) algorithms demonstrated the lowest mean integrated Brier score, whereas svm (0.515±0.085), glmboost (0.411±0.192), and blackboost (0.377±0.201) algorithms displayed the highest mean integrated Brier score. In the group of methods from the literature employing Boruta feature selection, penalized (penalized_B) (0.183±0.026) and ridge (ridge_B) (0.183±0.028) algorithms showcased the lowest mean integrated Brier score, while svm (svm_B) (0.515±0.068), glmboost (glmboost_B) (0.448±0.172), ctree (ctree_B) (0.418±0.077) and blackboost (blackboost_B) (0.416±0.182) algorithms presented the highest mean integrated Brier score. It was noted that among the voomStackLasso algorithms, the voomStackPrio2 and voomStackIPF4 exhibited the lowest mean integrated Brier score for ESCA data, at 0.205. This was followed by voomStackIPF1 (0.206±0.012) and voomStackIPF7 (0.206±0.013). The voomStackPrio1 algorithm (0.213±0.033) displayed the highest mean integrated Brier score.

Among the methods in the literature that applied Boruta feature selection, the mean number of selected features for ESCA data was the lowest (5.77±2.67). These were closely followed by the voomStackLasso algorithms (6.10±2.68). In terms of internal feature selection methods, the algorithm with the lowest mean number of features was obliqueRSF (774.70±585.68), while the algorithm with the highest mean number of features was rfsrc (1196.97±501.43).

The concordance index, integrated Brier score, and the number of selected features for Glioblastoma Multiforme (GBM) data are depicted in Figure 4.4, with related summary statistics presented in Table 4.4. After examining both the graph and the table for GBM data, it was observed that the ranger algorithm, when applied to

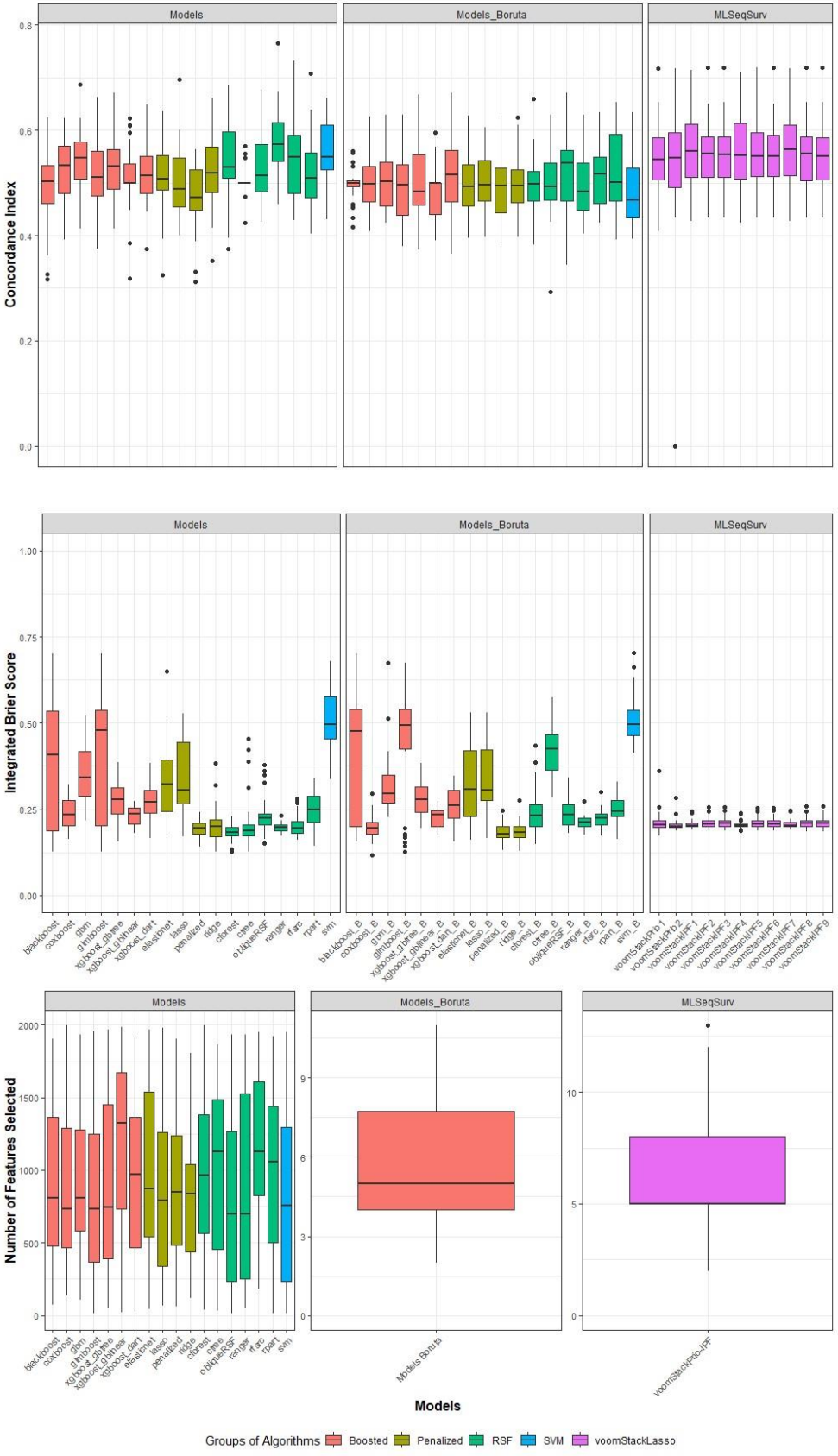


Figure 4.3. The concordance index, integrated Brier score, and the number of selected features for ESCA.

Table 4.3. The summary statistics of concordance index, integrated Brier score and the number of features selected for ESCA.

Groups of Algorithms	Models	Concordance Index			Integrated Brier Score			The Number of Features Selected		
		Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max
Models	**Blackboost**	0.493±0.075	0.503 (0.456-0.535)	0.317-0.625	0.377±0.201	0.408 (0.186-0.539)	0.127-0.702	897.37±528.96	809.00 (457.25-1419.00)	73-1903
	Cforest	0.541±0.074	0.530 (0.508-0.601)	0.374-0.685	**0.182±0.025**	0.183 (0.173-0.198)	0.126-0.229	1003.17±595.24	963.00 (560.25-1414.00)	38-1995
	Coxboost	0.524±0.062	0.534 (0.477-0.573)	0.392-0.622	0.240±0.047	0.235 (0.201-0.277)	0.163-0.323	910.03±563.47	734.00 (455.00-1351.50)	135-1998
	Ctree	0.500±0.028	0.500 (0.499-0.501)	0.424-0.569	0.211±0.080	0.187 (0.172-0.209)	0.127-0.455	1015.27±570.17	1129.50 (447.75-1501.00)	32-1864
	Elasticnet	0.510±0.068	0.508 (0.482-0.560)	0.325-0.636	0.338±0.113	0.322 (0.233-0.405)	0.172-0.649	962.00±576.62	871.50 (503.25-1588.50)	45-1965
	Gbm	0.540±0.060	0.547 (0.503-0.580)	0.412-0.687	0.356±0.085	0.342 (0.281-0.423)	0.218-0.520	884.97±534.81	808.50 (569.50-1392.50)	109-1933
	Glmboost	0.513±0.066	0.510 (0.472-0.561)	0.374-0.663	0.411±0.192	0.480 (0.201-0.548)	0.127-0.702	827.17±572.71	732.50 (352.00-1264.75)	15-1958
	Lasso	0.507±0.068	0.488 (0.454-0.550)	0.400-0.696	0.334±0.102	0.304 (0.263-0.445)	0.170-0.529	842.03±554.73	790.50 (322.00-1274.50)	68-1981
	ObliqueRSF	0.524±0.064	0.513 (0.480-0.577)	0.425-0.677	0.236±0.054	0.224 (0.204-0.241)	0.150-0.379	774.70±585.68	700.50 (219.75-1267.25)	13-1931
	Penalized	0.478±0.064	0.473 (0.447-0.529)	0.311-0.563	**0.193±0.025**	0.194 (0.177-0.212)	0.141-0.240	902.90±541.23	849.50 (454.75-1309.00)	59-1904
	Ranger	**0.580±0.060**	0.573 (0.538-0.617)	0.460-0.765	0.198±0.013	0.197 (0.188-0.205)	0.172-0.231	862.80±662.03	697.00 (237.50-1567.00)	47-1934
	Rfsrc	0.546±0.068	0.549 (0.476-0.591)	0.430-0.731	0.203±0.032	0.196 (0.180-0.215)	0.161-0.280	1196.97±501.43	1127.00 (792.00-1629.75)	181-1951
	Ridge	0.526±0.069	0.519 (0.477-0.569)	0.351-0.661	0.204±0.054	0.201 (0.168-0.225)	0.128-0.382	799.27±456.03	836.50 (412.75-1054.75)	119-1803
	Rpart	0.516±0.068	0.509 (0.466-0.559)	0.404-0.708	0.249±0.050	0.248 (0.208-0.290)	0.144-0.340	983.80±559.49	1058.50 (469.00-1457.00)	15-1922
	Svm	**0.559±0.057**	0.549 (0.521-0.613)	0.430-0.661	0.515±0.085	0.495 (0.450-0.578)	0.337-0.679	815.43±605.25	754.00 (208.25-1316.25)	12-1948
	Xgboost (dart)	0.520±0.063	0.513 (0.478-0.557)	0.375-0.649	0.272±0.052	0.270 (0.237-0.311)	0.165-0.384	924.87±560.28	972.00 (426.50-1422.00)	28-1908
	Xgboost (gblinear)	0.507±0.063	0.500 (0.495-0.540)	0.318-0.622	0.231±0.029	0.236 (0.200-0.255)	0.179-0.273	1190.40±597.32	1323.00 (682.50-1688.50)	23-1984
	Xgboost (gbtree)	0.529±0.060	0.532 (0.481-0.569)	0.413-0.672	0.276±0.058	0.278 (0.237-0.314)	0.155-0.386	913.70±617.37	746.50 (372.50-1498.25)	47-1969
Models Boruta	**Blackboost**	0.495±0.032	0.500 (0.487-0.505)	0.417-0.560	0.416±0.182	0.478 (0.195-0.541)	0.156-0.702	**5.77±2.67**	5.00 (4.00-8.00)	2-11
	Cforest	0.498±0.057	0.498 (0.461-0.527)	0.383-0.659	0.243±0.066	0.231 (0.198-0.268)	0.150-0.434			
	Coxboost	0.503±0.055	0.498 (0.463-0.531)	0.408-0.625	0.198±0.034	0.196 (0.176-0.214)	0.116-0.295			
	Ctree	0.500±0.066	0.492 (0.466-0.540)	0.292-0.629	0.418±0.077	0.425 (0.359-0.469)	0.283-0.575			
	Elasticnet	0.501±0.060	0.492 (0.452-0.539)	0.395-0.628	0.323±0.115	0.306 (0.211-0.424)	0.161-0.530			
	Gbm	0.502±0.053	0.503 (0.453-0.543)	0.423-0.630	0.324±0.091	0.296 (0.266-0.353)	0.227-0.675			
	Glmboost	0.493±0.065	0.496 (0.438-0.540)	0.379-0.630	0.448±0.172	0.494 (0.361-0.548)	0.128-0.675			
	Lasso	0.500±0.056	0.496 (0.458-0.545)	0.398-0.605	0.327±0.112	0.306 (0.263-0.427)	0.165-0.529			
	ObliqueRSF	**0.517±0.078**	0.537 (0.460-0.566)	0.344-0.670	0.240±0.046	0.233 (0.203-0.270)	0.180-0.343			
	Penalized	0.489±0.055	0.495 (0.438-0.533)	0.381-0.627	**0.183±0.026**	0.179 (0.168-0.201)	0.132-0.247			
	Ranger	0.492±0.062	0.484 (0.443-0.542)	0.403-0.628	0.214±0.019	0.213 (0.200-0.226)	0.177-0.273			
	Rfsrc	0.511±0.056	0.517 (0.459-0.550)	0.424-0.634	0.221±0.028	0.223 (0.201-0.236)	0.172-0.299			
	Ridge	0.497±0.057	0.495 (0.458-0.528)	0.396-0.624	**0.183±0.028**	0.182 (0.166-0.201)	0.128-0.276			
	Rpart	**0.519±0.076**	0.501 (0.464-0.599)	0.392-0.653	0.250±0.044	0.243 (0.228-0.277)	0.164-0.329			
	Svm	0.481±0.058	0.467 (0.431-0.532)	0.394-0.634	0.515±0.068	0.496 (0.464-0.547)	0.412-0.703			
	Xgboost (dart)	0.511±0.075	0.515 (0.457-0.569)	0.365-0.671	0.262±0.051	0.260 (0.221-0.306)	0.155-0.346			
	Xgboost (gblinear)	0.476±0.053	0.500 (0.434-0.501)	0.391-0.595	0.227±0.031	0.233 (0.196-0.249)	0.175-0.273			
	Xgboost (gbtree)	0.503±0.067	0.484 (0.453-0.558)	0.373-0.668	0.283±0.050	0.279 (0.240-0.319)	0.196-0.384			
MLSeqSurv	**voomStackPrio1**	0.550±0.065	0.544 (0.502-0.587)	0.408-0.717	0.213±0.033	0.205 (0.197-0.218)	0.173-0.362	**6.10±2.68**	5.00 (4.99-8.00)	2-13
	voomStackPrio2	0.530±0.122	0.548 (0.490-0.601)	0.000-0.717	0.205±0.018	0.201 (0.196-0.210)	0.187-0.283			
	voomStackIPF1	**0.558±0.064**	0.560 (0.509-0.614)	0.428-0.715	0.206±0.012	0.203 (0.200-0.211)	0.189-0.244			
	voomStackIPF2	0.552±0.062	0.555 (0.507-0.590)	0.434-0.718	0.210±0.016	0.208 (0.198-0.217)	0.187-0.256			
	voomStackIPF3	0.551±0.062	0.553 (0.506-0.589)	0.434-0.718	0.211±0.016	0.209 (0.199-0.218)	0.187-0.257			
	voomStackIPF4	0.553±0.065	0.553 (0.505-0.613)	0.424-0.711	0.205±0.011	0.203 (0.200-0.208)	0.187-0.240			
	voomStackIPF5	0.552±0.063	0.551 (0.507-0.597)	0.434-0.718	0.209±0.015	0.207 (0.198-0.217)	0.189-0.253			
	voomStackIPF6	0.551±0.063	0.551 (0.507-0.591)	0.434-0.718	0.210±0.016	0.208 (0.199-0.217)	0.189-0.254			
	voomStackIPF7	**0.560±0.063**	0.563 (0.513-0.617)	0.428-0.717	0.206±0.013	0.204 (0.199-0.212)	0.189-0.246			
	voomStackIPF8	0.551±0.064	0.556 (0.499-0.591)	0.434-0.718	0.211±0.016	0.210 (0.199-0.218)	0.186-0.258			
	voomStackIPF9	0.551±0.063	0.551 (0.501-0.588)	0.434-0.718	0.212±0.017	0.210 (0.199-0.219)	0.186-0.259			

internal feature selection, achieved the highest mean concordance index at 0.600. Within the category of methods applied to internal feature selection, the highest mean concordance indices were observed for ranger (0.600±0.045), cforest (0.593±0.052), gbm (0.590±0.060), penalized (0.585±0.056), elasticnet (0.583±0.060), lasso (0.582±0.052), coxboost (0.581±0.064), and rfsrc (0.581±0.050). Conversely, the lowest mean concordance index was attributed to the ctree algorithm (0.523±0.045). In the group of methods from the literature employing Boruta feature selection, the lasso (lasso_B) (0.577±0.040) and xgboost (with booster= "gblinear") (xgboost_gblinear_B) (0.575±0.042) algorithms demonstrated the highest mean concordance index, while the obliqueRSF (obliqueRSF_B) (0.528±0.068) and xgboost (with booster= "dart") (xgboost_dart_B) (0.530±0.047) algorithms exhibited the lowest mean concordance index. Among the voomStackLasso methods, the voomStackIPF1 (0.579±0.034) and voomStackPrio1 (0.576±0.042) algorithms showed the highest mean concordance index values, while the voomStackIPF7 algorithm (0.541±0.151) displayed the lowest mean concordance index.

It was observed that the cforest algorithm, when applied to internal feature selection, achieved the lowest mean integrated Brier score at 0.113 for GBM data. Within the category of methods applied to internal feature selection, the cforest (0.113±0.020) and penalized (0.115±0.022) algorithms demonstrated the lowest mean integrated Brier score, whereas svm (0.628±0.100), glmboost (0.501±0.277), and ctree (0.413±0.118) algorithms displayed the highest mean integrated Brier score. In the group of methods from the literature employing Boruta feature selection, the penalized (penalized_B) algorithm (0.117±0.020) showcased the lowest mean integrated Brier score, while the svm algorithm (0.632±0.133) presented the highest mean integrated Brier score. It was noted that among the voomStackLasso algorithms, the voomStackPrio5 and voomStackIPF6 exhibited the lowest mean integrated Brier score for GBM data, at 0.149. This was followed by voomStackIPF1, voomStackIPF2, voomStackIPF3, voomStackIPF4, and voomStackIPF7, all with a score of 0.150. The voomStackPrio1 algorithm (0.161±0.031) displayed the highest mean integrated Brier score.

Among the methods in the literature that applied Boruta feature selection, the mean number of selected features for GBM data was the lowest (7.87±3.16). These were closely followed by the voomStackLasso algorithms (7.90±4.07). In terms of

internal feature selection methods, the algorithm with the lowest mean number of features was ranger (647.80±478.69), while the algorithm with the highest mean number of features was xgboost (with booster= “gblinear”) (1330.83±561.83).

The concordance index, integrated Brier score, and the number of selected features for Kidney Renal Clear Cell Carcinoma (KIRC) data are depicted in Figure 4.5, with related summary statistics presented in Table 4.5. Upon reviewing both the graph and the table for KIRC data, it was observed that the cforest and ranger algorithms, when applied to internal feature selection, achieved the highest mean concordance index at 0.717. Following this, the rfsrc algorithm, with internal feature selection applied, yielded a mean concordance index of 0.708. Within the category of methods applied to internal feature selection, the highest mean concordance indices were observed for cforest (0.717±0.036), ranger (0.717±0.034), rfsrc (0.708±0.034), ridge (0.707±0.034), and blackboost (0.705±0.033) algorithms. Conversely, the lowest mean concordance index was attributed to the ctree algorithm (0.624±0.031). In the group of methods from the literature employing Boruta feature selection, the cforest (cforest_B) and ranger (ranger_B) algorithms demonstrated the highest mean concordance index at 0.693, while the svm algorithm (0.601±0.076) exhibited the lowest mean concordance index. Among the voomStackLasso methods, the voomStackIPF1, voomStackIPF4, and voomStackIPF7 algorithms showed the highest mean concordance index at 0.684, while the voomStackPrio2 algorithm (0.663±0.043) displayed the lowest mean concordance index.

It was observed that the cforest algorithm, when applied to internal feature selection, achieved the lowest mean integrated Brier score at 0.166 for KIRC data. Within the category of methods applied to internal feature selection, the cforest (0.166±0.009), penalized (0.171±0.012), ranger (0.173±0.007), and rfsrc (0.173±0.008) algorithms demonstrated the lowest mean integrated Brier score, whereas the svm algorithm (0.334±0.021) displayed the highest mean integrated Brier score. In the group of methods from the literature employing Boruta feature selection, penalized (0.168±0.015), ranger (0.173±0.011), coxboost (coxboost_B) (0.174±0.017), rfsrc (0.175±0.015) and cforest (cforest_B) (0.177±0.012) algorithms showcased the lowest mean integrated Brier score, while elasticnet (elasticnet_B) (0.413±0.162), svm (svm_B) (0.409±0.075), and lasso (lasso_B) (0.403±0.154) algorithms presented the highest mean integrated Brier score. It was noted that among

the voomStackLasso algorithms, the voomStackIPF2, voomStackIPF5, and voomStackIPF8 exhibited the lowest mean integrated Brier score for KIRC data, at 0.178. This was followed by voomStackIPF3, voomStackIPF6, voomStackIPF9, at 0.179. The voomStackPrio1 algorithm (0.193±0.029) displayed the highest mean integrated Brier score.

Among the methods in the voomStackLasso algorithms, the mean number of selected features for KIRC data was the lowest (30.03±10.82). These were closely followed by the methods in the literature that utilized Boruta feature selection (30.47±10.31). In terms of internal feature selection methods, the algorithm exhibiting the lowest mean number of features was rpart (763.03±503.42), whereas the algorithm demonstrating the highest mean number of features was xgboost (with booster= "gblinear") (1653.70±291.65).

The concordance index, integrated Brier score, and the number of selected features for Kidney Renal Papillary Cell Carcinoma (KIRP) data are depicted in Figure 4.6, with related summary statistics presented in Table 4.6. After examining both the graph and the table for KIRP data, it was observed that the rfsrc (rfsrc_B) algorithm, when applied to Boruta feature selection, achieved the highest mean concordance index at 0.818. This is followed by the blackboost (blackboost_B) algorithm, with Boruta feature selection applied, yielding a mean concordance index of 0.816. Among the methods employed for internal feature selection, the highest mean concordance indices were observed for lasso (0.805±0.075), elasticnet (0.804±0.068), and ranger (0.804±0.069). Conversely, the lowest mean concordance indices were attributed to ctree (0.719±0.103) and rpart (0.724±0.081) algorithms. In the group of methods from the literature employing Boruta feature selection, the rfsrc (rfsrc_B) (0.818±0.078), blackboost (blackboost_B) (0.816±0.072), and gbm (gbm_B) (0.812±0.074) algorithms demonstrated the highest mean concordance index, while the svm (svm_B) algorithm (0.607±0.261) exhibited the lowest mean concordance index. Among the voomStackLasso methods, the voomStackIPF7 algorithm (0.800±0.068) showed the highest mean concordance index, while the voomStackIPF6 algorithm (0.751±0.099) displayed the lowest mean concordance index.

It was noted that for KIRP data, the voomStackIPF8 algorithm attained the lowest mean integrated Brier score of 0.122. This was followed by voomStackIPF1

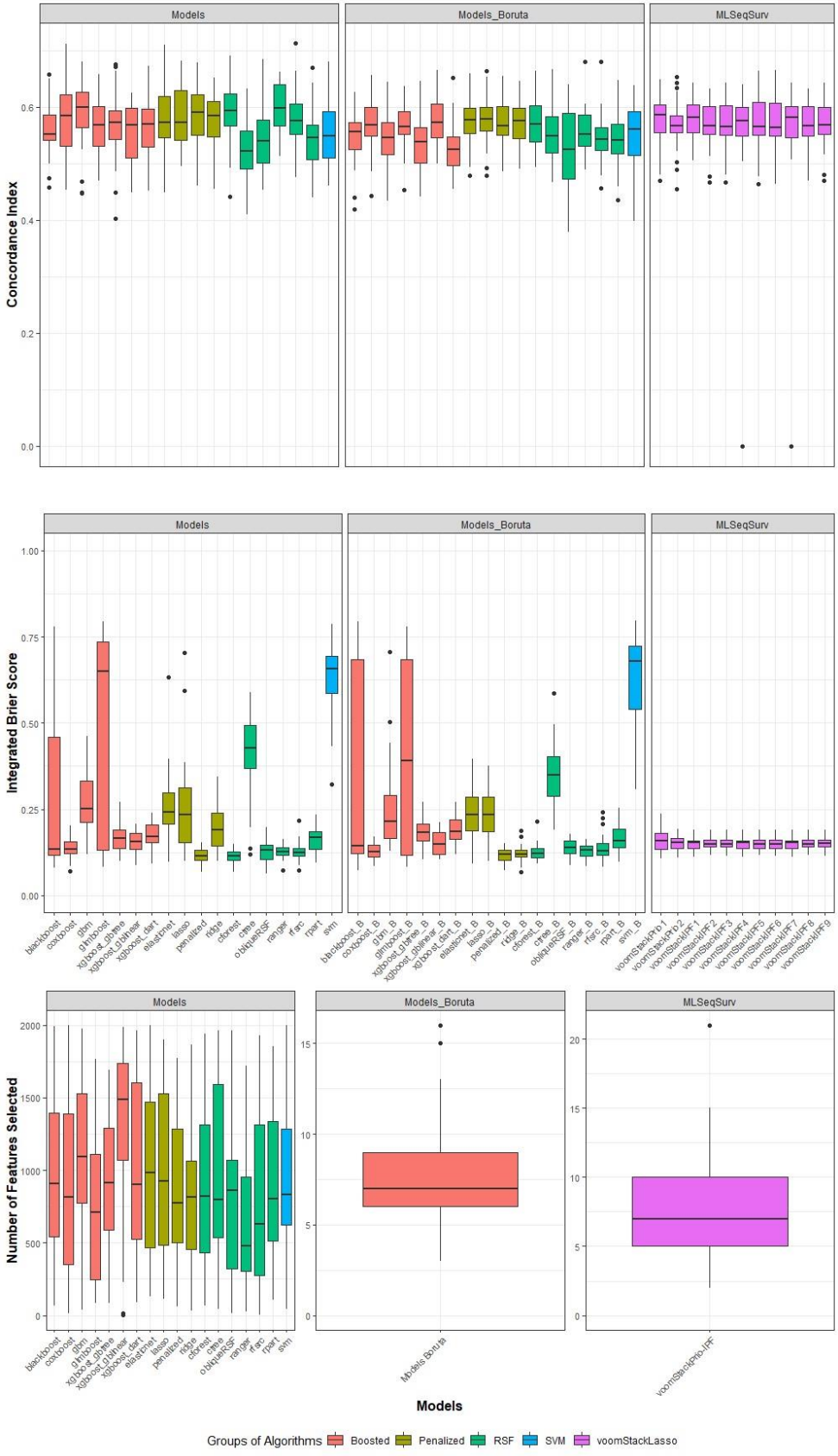


Figure 4.4. The concordance index, integrated Brier score, and the number of selected features for GBM.

Table 4.4. The summary statistics of concordance index, integrated Brier score and the number of features selected for GBM.

Groups of Algorithms	Models	Concordance Index			Integrated Brier Score			The Number of Features Selected		
		Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max
Models	**Blackboost**	0.561±0.047	0.552 (0.540-0.587)	0.458-0.657	0.283±0.256	0.134 (0.116-0.531)	0.081-0.780	987.83±529.60	910.50 (522.75-1400.25)	68-1990
	Cforest	**0.593±0.052**	0.594 (0.566-0.624)	0.441-0.690	**0.113±0.020**	0.115 (0.099-0.128)	0.068-0.148	891.90±601.21	819.50 (377.75-1354.00)	67-1941
	Coxboost	**0.581±0.064**	0.585 (0.531-0.625)	0.453-0.711	0.136±0.029	0.135 (0.121-0.160)	0.070-0.203	893.03±588.57	818.00 (290.25-1423.25)	16-1996
	Ctree	0.523±0.045	0.522 (0.491-0.559)	0.410-0.632	0.413±0.118	0.427 (0.362-0.499)	0.120-0.589	972.53±605.80	795.00 (516.50-1609.75)	41-1962
	Elasticnet	**0.583±0.060**	0.573 (0.543-0.622)	0.449-0.710	0.254±0.101	0.241 (0.206-0.299)	0.097-0.634	994.77±562.07	985.00 (446.00-1508.25)	130-2000
	Gbm	**0.590±0.060**	0.599 (0.561-0.630)	0.447-0.680	0.272±0.091	0.252 (0.205-0.339)	0.119-0.463	1121.17±545.80	1096.50 (734.00-1588.00)	40-1974
	Glmboost	0.568±0.050	0.568 (0.528-0.605)	0.469-0.658	0.501±0.277	0.649 (0.126-0.746)	0.083-0.794	742.83±522.26	709.50 (238.25-1121.25)	86-1764
	Lasso	**0.582±0.052**	0.572 (0.537-0.630)	0.495-0.682	0.259±0.135	0.235 (0.148-0.321)	0.099-0.704	987.37±533.94	926.50 (482.50-1549.25)	116-1898
	ObliqueRSF	0.543±0.056	0.541 (0.498-0.580)	0.453-0.684	0.129±0.032	0.132 (0.102-0.149)	0.064-0.197	798.77±543.87	861.50 (311.25-1098.25)	17-1966
	Penalized	**0.585±0.056**	0.591 (0.547-0.622)	0.461-0.679	**0.115±0.022**	0.115 (0.101-0.132)	0.067-0.155	866.83±502.49	774.00 (473.25-1287.00)	64-1775
	Ranger	**0.600±0.045**	0.598 (0.564-0.641)	0.513-0.662	0.127±0.019	0.126 (0.115-0.140)	0.073-0.162	647.80±478.69	476.50 (294.75-1021.75)	27-1722
	Rfsrc	**0.581±0.050**	0.576 (0.550-0.608)	0.476-0.712	0.126±0.027	0.123 (0.112-0.139)	0.073-0.218	792.90±608.56	632.00 (245.25-1351.25)	5-1927
	Ridge	0.578±0.051	0.585 (0.544-0.613)	0.455-0.652	0.201±0.073	0.190 (0.142-0.243)	0.100-0.343	839.80±545.74	814.50 (384.50-1093.75)	33-1864
	Rpart	0.544±0.050	0.546 (0.505-0.571)	0.439-0.670	0.165±0.036	0.169 (0.134-0.187)	0.096-0.234	869.53±474.88	804.00 (496.25-1350.50)	107-1853
	Svm	0.554±0.061	0.548 (0.509-0.594)	0.461-0.681	0.628±0.100	0.657 (0.583-0.695)	0.323-0.786	937.73±537.22	835.00 (600.50-1366.50)	45-1996
	Xgboost (dart)	0.564±0.048	0.570 (0.524-0.600)	0.452-0.673	0.174±0.040	0.170 (0.151-0.207)	0.093-0.240	1025.43±598.77	901.50 (491.25-1624.25)	90-1965
	Xgboost (gblinear)	0.555±0.051	0.569 (0.503-0.599)	0.449-0.625	0.153±0.032	0.157 (0.133-0.181)	0.088-0.207	1330.83±561.83	1486.00 (1047.25-1738.50)	2-1988
	Xgboost (gbtree)	0.567±0.057	0.573 (0.540-0.594)	0.403-0.675	0.170±0.041	0.165 (0.135-0.193)	0.100-0.271	908.07±466.79	911.00 (528.75-1334.25)	85-1690
Models Boruta	**Blackboost**	0.548±0.046	0.556 (0.524-0.576)	0.419-0.626	0.347±0.279	0.145 (0.120-0.700)	0.073-0.794	**7.87±3.16**	7.00 (6.00-9.00)	3-16
	Cforest	0.571±0.043	0.569 (0.537-0.603)	0.494-0.663	0.127±0.024	0.122 (0.109-0.138)	0.094-0.214			
	Coxboost	0.567±0.045	0.569 (0.546-0.600)	0.443-0.657	0.127±0.024	0.126 (0.109-0.147)	0.085-0.171			
	Ctree	0.549±0.047	0.548 (0.515-0.587)	0.467-0.667	0.353±0.094	0.349 (0.280-0.409)	0.190-0.587			
	Elasticnet	0.573±0.043	0.578 (0.551-0.599)	0.479-0.659	0.235±0.079	0.235 (0.178-0.289)	0.092-0.395			
	Gbm	0.542±0.050	0.546 (0.514-0.575)	0.434-0.644	0.252±0.129	0.215 (0.166-0.304)	0.130-0.705			
	Glmboost	0.565±0.041	0.565 (0.550-0.594)	0.453-0.636	0.397±0.286	0.390 (0.115-0.688)	0.082-0.780			
	Lasso	**0.577±0.040**	0.578 (0.557-0.600)	0.479-0.664	0.234±0.080	0.234 (0.178-0.292)	0.099-0.377			
	ObliqueRSF	0.528±0.068	0.525 (0.471-0.594)	0.378-0.641	0.137±0.024	0.138 (0.120-0.159)	0.087-0.179			
	Penalized	0.572±0.040	0.567 (0.547-0.602)	0.486-0.654	**0.117±0.020**	0.120 (0.101-0.132)	0.073-0.152			
	Ranger	0.557±0.041	0.551 (0.529-0.589)	0.489-0.680	0.129±0.019	0.131 (0.115-0.144)	0.086-0.162			
	Rfsrc	0.546±0.044	0.543 (0.523-0.565)	0.457-0.680	0.138±0.036	0.129 (0.116-0.152)	0.083-0.242			
	Ridge	0.569±0.039	0.575 (0.541-0.599)	0.491-0.646	0.122±0.025	0.119 (0.108-0.134)	0.069-0.189			
	Rpart	0.543±0.049	0.542 (0.515-0.572)	0.436-0.648	0.165±0.039	0.159 (0.138-0.198)	0.097-0.253			
	Svm	0.548±0.066	0.561 (0.511-0.596)	0.398-0.639	0.632±0.133	0.679 (0.522-0.731)	0.308-0.796			
	Xgboost (dart)	0.530±0.047	0.525 (0.495-0.549)	0.455-0.651	0.190±0.041	0.186 (0.162-0.226)	0.119-0.270			
	Xgboost (gblinear)	**0.575±0.042**	0.573 (0.542-0.605)	0.500-0.665	0.152±0.036	0.149 (0.118-0.184)	0.105-0.213			
	Xgboost (gbtree)	0.535±0.047	0.539 (0.500-0.566)	0.441-0.645	0.188±0.045	0.182 (0.156-0.214)	0.105-0.272			
MLSeqSurv	**voomStackPrio1**	**0.576±0.042**	0.586 (0.554-0.607)	0.469-0.649	0.161±0.031	0.158 (0.133-0.183)	0.106-0.236	**7.90±4.07**	7.00 (4.75-10.00)	2-21
	voomStackPrio2	0.569±0.045	0.566 (0.552-0.593)	0.455-0.654	**0.153±0.021**	0.154 (0.136-0.166)	0.109-0.193			
	voomStackIPF1	**0.579±0.034**	0.581 (0.553-0.605)	0.505-0.643	**0.150±0.017**	0.152 (0.136-0.160)	0.112-0.190			
	voomStackIPF2	**0.572±0.042**	0.567 (0.552-0.605)	0.467-0.632	**0.150±0.017**	0.147 (0.140-0.163)	0.116-0.191			
	voomStackIPF3	**0.570±0.042**	0.565 (0.549-0.605)	0.467-0.642	**0.150±0.017**	0.148 (0.141-0.163)	0.115-0.191			
	voomStackIPF4	0.557±0.110	0.575 (0.547-0.601)	0.000-0.641	**0.150±0.017**	0.152 (0.136-0.160)	0.112-0.190			
	voomStackIPF5	**0.572±0.046**	0.565 (0.549-0.613)	0.463-0.663	**0.149±0.017**	0.147 (0.136-0.161)	0.116-0.191			
	voomStackIPF6	**0.572±0.046**	0.564 (0.547-0.613)	0.463-0.665	**0.149±0.017**	0.148 (0.136-0.161)	0.115-0.191			
	voomStackIPF7	0.541±0.151	0.581 (0.544-0.602)	0.000-0.643	**0.150±0.017**	0.152 (0.136-0.160)	0.112-0.190			
	voomStackIPF8	**0.571±0.041**	0.567 (0.547-0.602)	0.470-0.632	**0.151±0.017**	0.149 (0.141-0.163)	0.116-0.191			
	voomStackIPF9	**0.571±0.042**	0.568 (0.551-0.602)	0.470-0.642	**0.151±0.017**	0.150 (0.141-0.163)	0.115-0.191			

(0.125±0.018) and voomStackIPF7 (0.125±0.018). The voomStackPrio1 algorithm (0.189±0.087) displayed the highest mean integrated Brier score. Within the category of methods applied to internal feature selection, the penalized (0.140±0.022), cforest (0.142±0.022), and ranger (0.143±0.017) algorithms demonstrated the lowest mean integrated Brier score, whereas the elasticnet algorithm (0.606±0.162) displayed the highest mean integrated Brier score. In the category of methods from the literature utilizing Boruta feature selection, the penalized (penalized_B) and ranger (ranger_B) algorithms showcased the lowest mean integrated Brier score at 0.146, while elasticnet (elasticnet_B) (0.670±0.060) and lasso (lasso_B) (0.657±0.104) algorithms presented the highest mean integrated Brier score.

Among the methods in the literature that applied Boruta feature selection, the mean number of selected features for KIRP data was the lowest (34.40±6.41). These were closely followed by the voomStackLasso algorithms (38.90±8.30). In terms of internal feature selection methods, the algorithm with the lowest mean number of features was rpart (771.13±591.05), while the algorithm with the highest mean number of features was xgboost (with booster= “gblinear”) (1485.13±534.78).

The concordance index, integrated Brier score, and the number of selected features for Acute Myeloid Leukemia (LAML) data are depicted in Figure 4.7, with related summary statistics presented in Table 4.7. After examining both the graph and the table for LAML data, it was observed that the coxboost and rfsrc algorithms, when applied to internal feature selection, achieved the highest mean concordance index at 0.667. This is followed by the elasticnet (0.664±0.065), lasso (0.664±0.070), cforest (0.662±0.063), ranger (0.662±0.055), and ridge (0.660±0.059) algorithms, where internal feature selection was applied, resulting in a mean concordance index. Meanwhile, the ctree algorithm (0.554±0.052) exhibited the lowest mean concordance index. Among the voomStackLasso methods, the voomStackIPF7 algorithm (0.640±0.052) showed the highest mean concordance index, while the voomStackIPF6 algorithm (0.592±0.086) displayed the lowest mean concordance index. In the group of methods from the literature employing Boruta feature selection, the ranger (ranger_B) (0.629±0.048) and rfsrc (rfsrc_B) (0.622±0.058) algorithms demonstrated the highest mean concordance index, while the svm (svm_B) algorithm (0.527±0.085) exhibited the lowest mean concordance index.

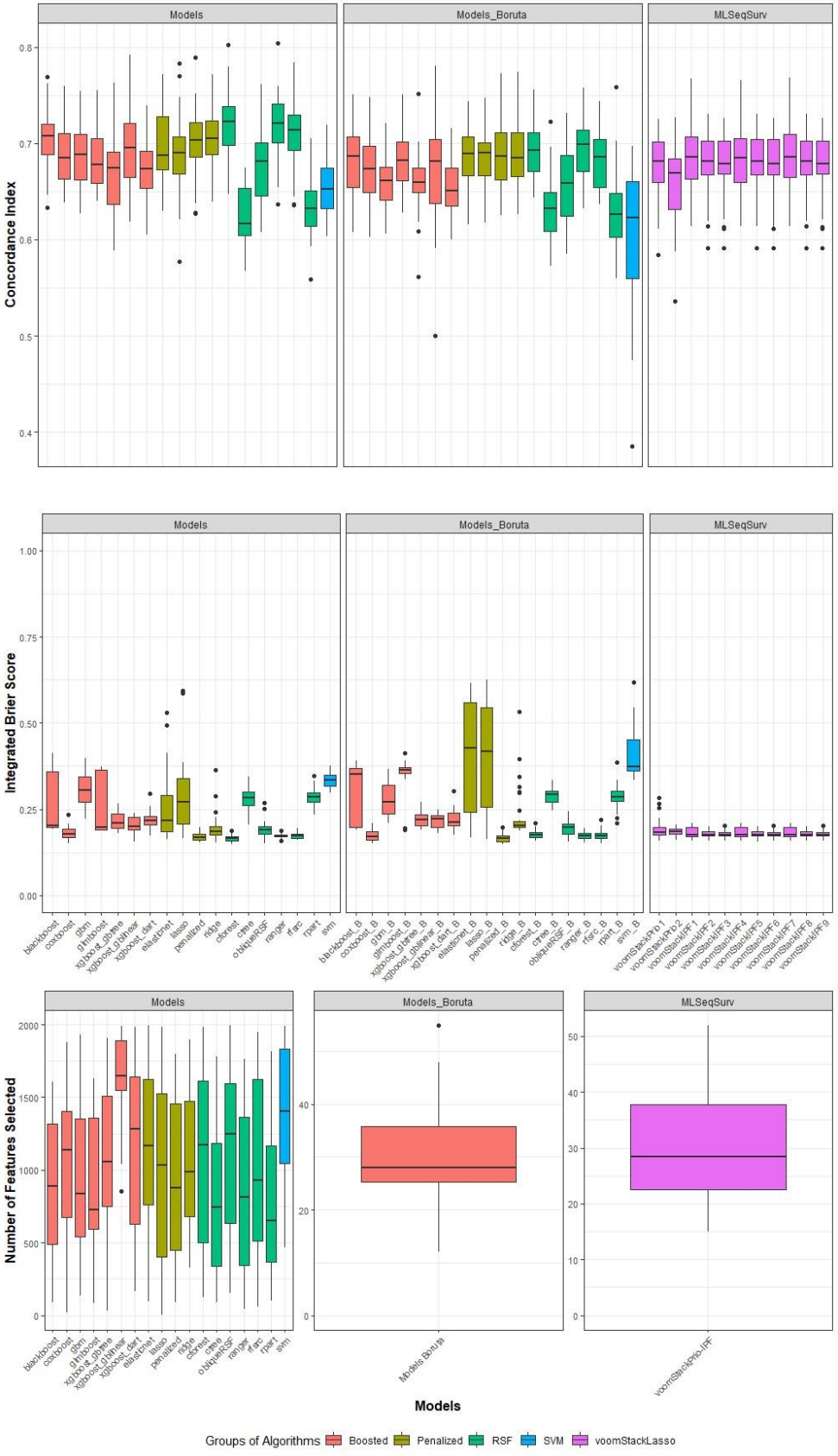


Figure 4.5. The concordance index, integrated Brier score, and the number of selected features for KIRC.

Table 4.5. The summary statistics of concordance index, integrated Brier score and the number of features selected for KIRC.

Groups of Algorithms	Models	Concordance Index			Integrated Brier Score			The Number of Features Selected		
		Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max
Models	**Blackboost**	0.705±0.033	0.708 (0.687-0.721)	0.634-0.769	0.255±0.081	0.203 (0.197-0.359)	0.193-0.413	905.80±471.27	887.50 (459.25-1327.25)	91-1604
	Cforest	**0.717±0.036**	0.722 (0.696-0.740)	0.647-0.802	**0.166±0.009**	0.165 (0.159-0.171)	0.150-0.187	1086.53±587.94	1175.00 (425.50-1635.50)	125-1983
	Coxboost	0.688±0.032	0.685 (0.663-0.715)	0.638-0.759	0.180±0.021	0.178 (0.166-0.193)	0.151-0.234	1047.83±502.49	1135.00 (650.25-1411.00)	21-1879
	Ctree	0.624±0.031	0.617 (0.603-0.654)	0.567-0.674	0.279±0.034	0.283 (0.261-0.301)	0.206-0.344	783.13±511.44	746.00 (315.00-1194.75)	89-1781
	Elasticnet	0.697±0.034	0.688 (0.672-0.729)	0.629-0.772	0.254±0.096	0.217 (0.185-0.297)	0.163-0.531	1194.97±537.78	1165.50 (720.25-1634.00)	95-1992
	Gbm	0.686±0.032	0.689 (0.660-0.711)	0.627-0.754	0.308±0.048	0.305 (0.266-0.348)	0.222-0.397	960.50±509.20	835.50 (528.00-1380.00)	139-1928
	Glmboost	0.684±0.033	0.677 (0.657-0.707)	0.640-0.755	0.270±0.086	0.197 (0.189-0.364)	0.187-0.374	882.87±495.71	729.50 (570.00-1396.50)	84-1627
	Lasso	0.689±0.044	0.690 (0.667-0.707)	0.577-0.783	0.288±0.104	0.271 (0.206-0.342)	0.166-0.595	984.90±615.60	1035.00 (338.25-1535.50)	2-1984
	ObliqueRSF	0.674±0.038	0.681 (0.645-0.703)	0.608-0.761	0.191±0.024	0.189 (0.179-0.200)	0.152-0.269	1111.73±581.12	1250.50 (606.00-1611.25)	152-1996
	Penalized	0.701±0.037	0.704 (0.683-0.723)	0.627-0.789	**0.171±0.012**	0.167 (0.161-0.178)	0.153-0.198	929.70±563.54	876.50 (426.75-1469.25)	91-1797
	Ranger	**0.717±0.034**	0.720 (0.700-0.742)	0.637-0.804	**0.173±0.007**	0.172 (0.169-0.177)	0.158-0.188	863.50±557.10	814.50 (329.50-1401.50)	44-1761
	Rfsrc	0.708±0.034	0.714 (0.692-0.731)	0.636-0.784	**0.173±0.008**	0.173 (0.165-0.178)	0.162-0.194	1000.00±625.51	929.50 (470.00-1638.50)	63-1948
	Ridge	0.707±0.034	0.705 (0.688-0.726)	0.639-0.772	0.197±0.041	0.185 (0.175-0.202)	0.154-0.364	1049.70±496.30	988.50 (674.25-1498.75)	328-1896
	Rpart	0.635±0.034	0.632 (0.614-0.654)	0.559-0.705	0.284±0.024	0.285 (0.269-0.299)	0.235-0.347	763.03±503.42	653.50 (362.50-1174.50)	102-1813
	Svm	0.655±0.032	0.653 (0.630-0.674)	0.603-0.719	0.334±0.021	0.333 (0.315-0.351)	0.297-0.377	1367.20±484.02	1405.50 (997.25-1850.75)	468-1986
	Xgboost (dart)	0.673±0.034	0.673 (0.650-0.693)	0.605-0.740	0.218±0.024	0.218 (0.203-0.232)	0.174-0.295	1131.07±608.32	1282.00 (583.75-1673.75)	167-1985
	Xgboost (gblinear)	0.696±0.039	0.695 (0.661-0.722)	0.618-0.792	0.204±0.021	0.201 (0.189-0.227)	0.155-0.240	1653.70±291.65	1649.50 (1536.50-1904.50)	857-1986
	Xgboost (gbtree)	0.670±0.042	0.674 (0.636-0.697)	0.589-0.763	0.216±0.025	0.211 (0.195-0.240)	0.180-0.265	1050.07±558.69	1058.50 (705.75-1524.00)	34-1907
Models Boruta	**Blackboost**	0.681±0.036	0.687 (0.652-0.708)	0.608-0.751	0.297±0.085	0.352 (0.197-0.369)	0.190-0.392	**30.47±10.31**	28.00 (24.50-36.25)	12-55
	Cforest	**0.693±0.030**	0.693 (0.669-0.712)	0.644-0.756	**0.177±0.012**	0.175 (0.169-0.184)	0.158-0.210			
	Coxboost	0.673±0.034	0.673 (0.648-0.700)	0.603-0.748	**0.174±0.017**	0.171 (0.160-0.186)	0.152-0.210			
	Ctree	0.631±0.032	0.633 (0.607-0.651)	0.572-0.722	0.288±0.022	0.292 (0.270-0.304)	0.247-0.335			
	Elasticnet	0.686±0.032	0.689 (0.665-0.709)	0.615-0.744	0.413±0.162	0.427 (0.237-0.567)	0.169-0.617			
	Gbm	0.658±0.026	0.661 (0.640-0.676)	0.606-0.721	0.274±0.047	0.270 (0.233-0.322)	0.210-0.367			
	Glmboost	0.683±0.031	0.682 (0.660-0.701)	0.628-0.750	0.348±0.055	0.364 (0.354-0.371)	0.189-0.412			
	Lasso	0.685±0.030	0.690 (0.666-0.702)	0.617-0.747	0.403±0.154	0.416 (0.252-0.556)	0.164-0.626			
	ObliqueRSF	0.658±0.041	0.659 (0.619-0.689)	0.585-0.731	0.196±0.022	0.197 (0.176-0.208)	0.157-0.243			
	Penalized	0.689±0.036	0.687 (0.662-0.713)	0.626-0.772	**0.168±0.015**	0.167 (0.156-0.174)	0.148-0.197			
	Ranger	**0.693±0.031**	0.699 (0.668-0.716)	0.632-0.758	**0.173±0.011**	0.174 (0.166-0.179)	0.153-0.195			
	Rfsrc	0.683±0.030	0.686 (0.651-0.705)	0.637-0.744	**0.175±0.015**	0.172 (0.164-0.181)	0.152-0.219			
	Ridge	0.690±0.035	0.685 (0.665-0.713)	0.626-0.774	0.234±0.077	0.203 (0.196-0.225)	0.188-0.534			
	Rpart	0.630±0.041	0.626 (0.602-0.649)	0.560-0.758	0.289±0.035	0.285 (0.274-0.306)	0.211-0.387			
	Svm	0.601±0.076	0.623 (0.553-0.663)	0.386-0.697	0.409±0.075	0.373 (0.361-0.467)	0.334-0.617			
	Xgboost (dart)	0.653±0.032	0.651 (0.632-0.676)	0.600-0.715	0.221±0.029	0.213 (0.200-0.240)	0.175-0.304			
	Xgboost (gblinear)	0.657±0.082	0.681 (0.632-0.707)	0.500-0.780	0.216±0.021	0.221 (0.196-0.232)	0.181-0.248			
	Xgboost (gbtree)	0.659±0.032	0.660 (0.648-0.675)	0.561-0.751	0.222±0.022	0.220 (0.201-0.237)	0.191-0.270			
MLSeqSurv	**voomStackPrio1**	0.675±0.035	0.681 (0.654-0.702)	0.584-0.725	0.193±0.029	0.183 (0.176-0.199)	0.159-0.284	**30.03±10.82**	28.50 (21.25-38.75)	15-52
	voomStackPrio2	0.663±0.043	0.669 (0.628-0.688)	0.536-0.727	0.185±0.010	0.186 (0.178-0.192)	0.160-0.206			
	voomStackIPF1	**0.684±0.036**	0.685 (0.658-0.710)	0.614-0.767	0.181±0.015	0.176 (0.170-0.198)	0.157-0.209			
	voomStackIPF2	0.676±0.034	0.682 (0.661-0.703)	0.591-0.730	**0.178±0.011**	0.176 (0.172-0.187)	0.159-0.200			
	voomStackIPF3	0.674±0.034	0.679 (0.662-0.703)	0.592-0.726	**0.179±0.011**	0.176 (0.172-0.185)	0.159-0.202			
	voomStackIPF4	**0.684±0.036**	0.685 (0.656-0.708)	0.614-0.766	0.181±0.015	0.176 (0.170-0.198)	0.157-0.209			
	voomStackIPF5	0.677±0.034	0.682 (0.661-0.705)	0.591-0.730	**0.178±0.011**	0.176 (0.172-0.187)	0.157-0.200			
	voomStackIPF6	0.675±0.034	0.679 (0.662-0.704)	0.592-0.726	**0.179±0.011**	0.176 (0.172-0.185)	0.157-0.202			
	voomStackIPF7	**0.684±0.036**	0.686 (0.658-0.713)	0.614-0.768	0.181±0.015	0.176 (0.170-0.198)	0.157-0.209			
	voomStackIPF8	0.676±0.034	0.682 (0.661-0.703)	0.591-0.730	**0.178±0.011**	0.176 (0.172-0.187)	0.159-0.200			
	voomStackIPF9	0.674±0.034	0.679 (0.662-0.703)	0.592-0.726	**0.179±0.011**	0.176 (0.172-0.185)	0.159-0.202			

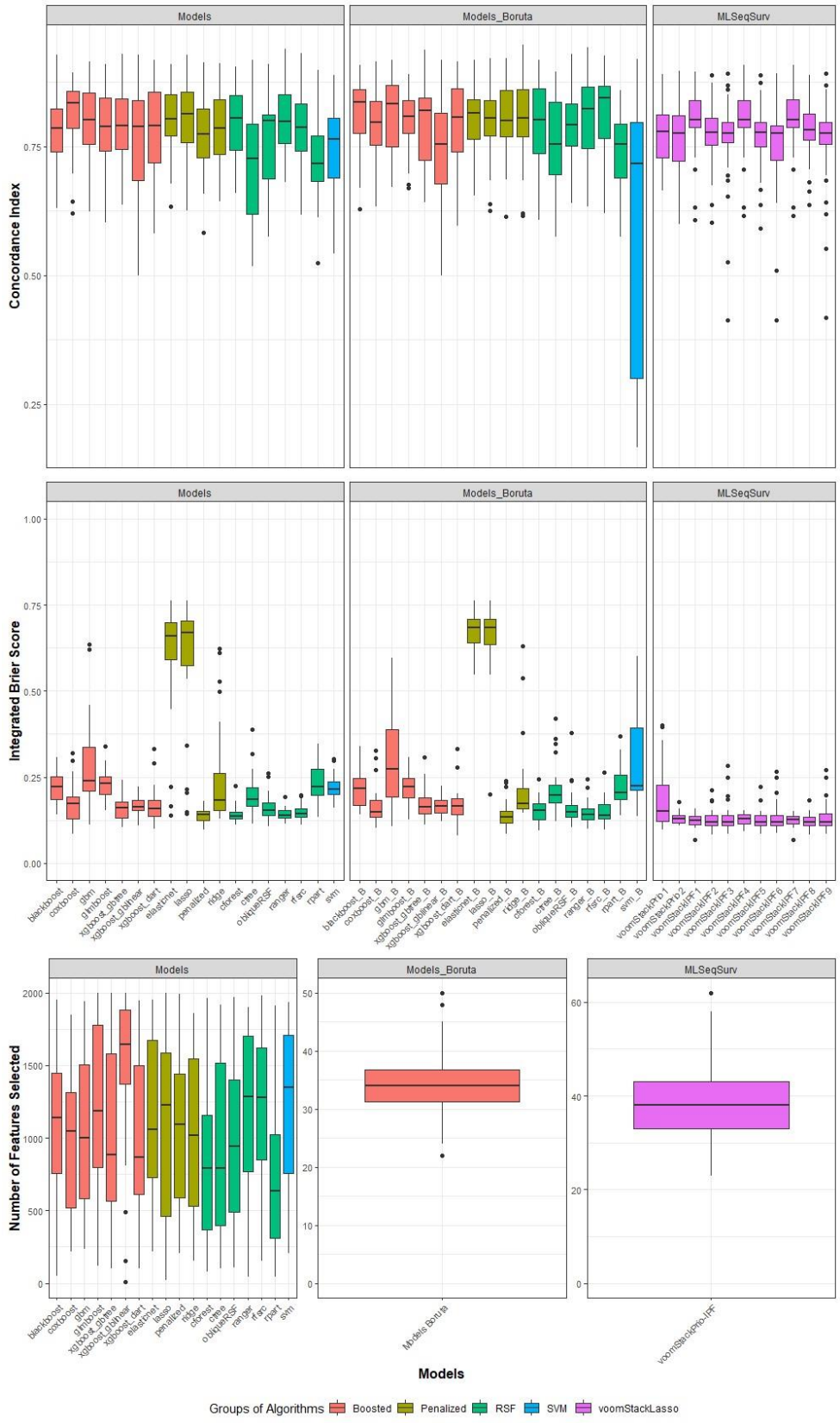


Figure 4.6. The concordance index, integrated Brier score, and the number of selected features for KIRP.

Table 4.6. The summary statistics of concordance index, integrated Brier score and the number of features selected for KIRP.

Groups of Algorithms	Models	Concordance Index			Integrated Brier Score			The Number of Features Selected		
		Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max
Models	**Blackboost**	0.787±0.070	0.785 (0.735-0.825)	0.631-0.927	0.218±0.046	0.222 (0.179-0.253)	0.143-0.307	1110.47±482.44	1139.50 (709.00-1459.75)	50-1954
	Cforest	0.791±0.072	0.805 (0.739-0.851)	0.659-0.905	**0.142±0.022**	0.137 (0.128-0.150)	0.113-0.224	846.97±531.93	793.00 (352.00-1186.25)	79-1965
	Coxboost	**0.812±0.071**	0.835 (0.778-0.859)	0.619-0.893	0.174±0.059	0.173 (0.127-0.197)	0.085-0.321	987.37±469.36	1050.00 (492.50-1340.25)	220-1848
	Ctree	0.719±0.103	0.727 (0.618-0.800)	0.518-0.918	0.199±0.058	0.186 (0.164-0.228)	0.113-0.388	927.80±573.74	794.00 (373.75-1562.25)	100-1916
	Elasticnet	**0.804±0.068**	0.804 (0.768-0.855)	0.632-0.910	0.606±0.162	0.661 (0.586-0.703)	0.138-0.762	1124.47±524.04	1057.00 (716.25-1684.75)	215-1952
	Gbm	0.797±0.069	0.802 (0.752-0.857)	0.623-0.914	0.282±0.129	0.239 (0.207-0.348)	0.112-0.635	1057.20±519.82	1000.00 (560.75-1523.25)	234-1940
	Glmboost	0.788±0.075	0.789 (0.738-0.846)	0.603-0.910	0.228±0.043	0.232 (0.197-0.253)	0.154-0.339	1245.53±582.77	1189.00 (728.50-1802.00)	119-2000
	Lasso	**0.805±0.075**	0.813 (0.752-0.858)	0.624-0.927	0.590±0.186	0.670 (0.560-0.707)	0.143-0.762	1053.20±642.58	1229.00 (442.00-1624.75)	23-1997
	ObliqueRSF	0.764±0.087	0.799 (0.684-0.812)	0.574-0.909	0.161±0.036	0.154 (0.137-0.179)	0.106-0.260	957.30±580.57	940.50 (452.50-1431.25)	107-1972
	Penalized	0.774±0.076	0.774 (0.727-0.828)	0.583-0.912	**0.140±0.022**	0.142 (0.122-0.153)	0.098-0.181	1048.40±526.04	1094.50 (572.50-1486.00)	205-1990
	Ranger	**0.804±0.069**	0.798 (0.755-0.853)	0.681-0.939	**0.143±0.017**	0.139 (0.132-0.153)	0.113-0.192	1176.30±550.23	1286.50 (741.50-1738.50)	41-1898
	Rfsrc	0.787±0.077	0.787 (0.735-0.833)	0.618-0.930	0.149±0.022	0.145 (0.134-0.159)	0.113-0.198	1227.97±502.89	1278.50 (819.25-1637.00)	155-1980
	Ridge	0.788±0.078	0.786 (0.729-0.843)	0.642-0.912	0.248±0.145	0.182 (0.152-0.273)	0.128-0.622	1022.23±567.98	1018.50 (495.50-1565.75)	153-1861
	Rpart	0.724±0.081	0.717 (0.678-0.774)	0.524-0.898	0.232±0.057	0.223 (0.190-0.286)	0.133-0.346	771.13±591.05	633.00 (296.75-1080.00)	41-1912
	Svm	0.747±0.077	0.764 (0.688-0.806)	0.542-0.888	0.220±0.036	0.216 (0.199-0.238)	0.162-0.303	1250.50±509.73	1347.00 (738.50-1715.00)	208-1935
	Xgboost (dart)	0.782±0.090	0.790 (0.715-0.857)	0.580-0.918	0.170±0.050	0.159 (0.135-0.184)	0.100-0.332	1020.50±553.27	865.50 (606.75-1544.00)	99-1947
	Xgboost (gblinear)	0.761±0.117	0.788 (0.683-0.842)	0.500-0.928	0.167±0.026	0.164 (0.152-0.183)	0.110-0.223	1485.13±534.78	1644.50 (1323.00-1883.00)	11-1999
	Xgboost (gbtree)	0.785±0.074	0.791 (0.742-0.847)	0.637-0.929	0.162±0.039	0.161 (0.131-0.183)	0.104-0.242	1007.60±605.46	884.00 (493.00-1598.75)	100-1996
Models Boruta	**Blackboost**	**0.816±0.072**	0.837 (0.770-0.860)	0.629-0.907	0.218±0.049	0.216 (0.166-0.250)	0.142-0.339	**34.40±6.41**	34.00 (31.00-37.50)	22-50
	Cforest	0.797±0.087	0.802 (0.731-0.868)	0.606-0.917	0.153±0.033	0.154 (0.127-0.175)	0.094-0.244			
	Coxboost	0.795±0.070	0.796 (0.749-0.842)	0.632-0.914	0.167±0.054	0.148 (0.131-0.188)	0.102-0.327			
	Ctree	0.754±0.092	0.754 (0.692-0.837)	0.574-0.895	0.214±0.067	0.197 (0.174-0.228)	0.123-0.419			
	Elasticnet	**0.802±0.068**	0.815 (0.760-0.844)	0.655-0.917	0.670±0.060	0.683 (0.628-0.712)	0.548-0.762			
	Gbm	**0.812±0.074**	0.832 (0.742-0.870)	0.671-0.917	0.310±0.143	0.273 (0.186-0.421)	0.108-0.596			
	Glmboost	0.799±0.059	0.809 (0.770-0.841)	0.670-0.889	0.218±0.046	0.222 (0.184-0.248)	0.126-0.307			
	Lasso	0.799±0.074	0.805 (0.767-0.844)	0.624-0.921	0.657±0.104	0.684 (0.624-0.712)	0.199-0.762			
	ObliqueRSF	0.789±0.069	0.792 (0.749-0.835)	0.640-0.930	0.162±0.053	0.148 (0.134-0.170)	0.105-0.379			
	Penalized	**0.804±0.074**	0.801 (0.768-0.862)	0.614-0.920	**0.146±0.046**	0.133 (0.115-0.160)	0.085-0.238			
	Ranger	**0.803±0.083**	0.823 (0.745-0.868)	0.632-0.942	**0.146±0.033**	0.142 (0.127-0.159)	0.101-0.245			
	Rfsrc	**0.818±0.078**	0.844 (0.762-0.868)	0.619-0.926	0.149±0.035	0.139 (0.127-0.172)	0.097-0.263			
	Ridge	**0.805±0.082**	0.804 (0.766-0.867)	0.616-0.946	0.226±0.126	0.174 (0.157-0.221)	0.145-0.630			
	Rpart	0.741±0.079	0.755 (0.681-0.796)	0.574-0.858	0.224±0.057	0.206 (0.183-0.258)	0.138-0.368			
	Svm	0.607±0.261	0.717 (0.291-0.807)	0.166-0.920	0.300±0.135	0.225 (0.212-0.412)	0.136-0.602			
	Xgboost (dart)	0.797±0.082	0.807 (0.736-0.865)	0.595-0.913	0.167±0.048	0.166 (0.139-0.188)	0.080-0.333			
	Xgboost (gblinear)	0.725±0.135	0.754 (0.662-0.823)	0.500-0.918	0.164±0.024	0.167 (0.145-0.182)	0.121-0.224			
	Xgboost (gbtree)	0.791±0.080	0.820 (0.715-0.844)	0.641-0.937	0.172±0.046	0.163 (0.140-0.192)	0.113-0.308			
MLSeqSurv	**voomStackPrio1**	0.774±0.062	0.779 (0.721-0.817)	0.664-0.890	0.189±0.087	0.152 (0.119-0.232)	0.098-0.402	**38.90±8.30**	38.00 (33.00-43.00)	23-62
	voomStackPrio2	0.764±0.075	0.775 (0.719-0.811)	0.599-0.896	0.131±0.017	0.129 (0.118-0.140)	0.109-0.177			
	voomStackIPF1	0.799±0.068	0.801 (0.786-0.848)	0.606-0.895	**0.125±0.018**	0.125 (0.113-0.139)	0.069-0.159			
	voomStackIPF2	0.773±0.065	0.778 (0.749-0.809)	0.603-0.888	**0.127±0.028**	0.120 (0.109-0.141)	0.083-0.212			
	voomStackIPF3	0.759±0.095	0.775 (0.756-0.801)	0.413-0.891	0.133±0.043	0.118 (0.110-0.143)	0.086-0.283			
	voomStackIPF4	0.799±0.068	0.801 (0.785-0.848)	0.616-0.908	**0.127±0.016**	0.128 (0.113-0.143)	0.091-0.153			
	voomStackIPF5	0.769±0.066	0.778 (0.746-0.799)	0.591-0.888	**0.128±0.029**	0.120 (0.110-0.141)	0.085-0.222			
	voomStackIPF6	0.751±0.099	0.775 (0.713-0.793)	0.413-0.891	0.133±0.042	0.120 (0.110-0.143)	0.087-0.267			
	voomStackIPF7	**0.800±0.068**	0.802 (0.786-0.847)	0.616-0.908	**0.125±0.018**	0.126 (0.113-0.139)	0.069-0.152			
	voomStackIPF8	0.780±0.058	0.783 (0.758-0.817)	0.637-0.888	**0.122±0.021**	0.119 (0.109-0.135)	0.083-0.183			
	voomStackIPF9	0.755±0.097	0.775 (0.742-0.801)	0.417-0.891	0.134±0.042	0.120 (0.110-0.144)	0.085-0.272			

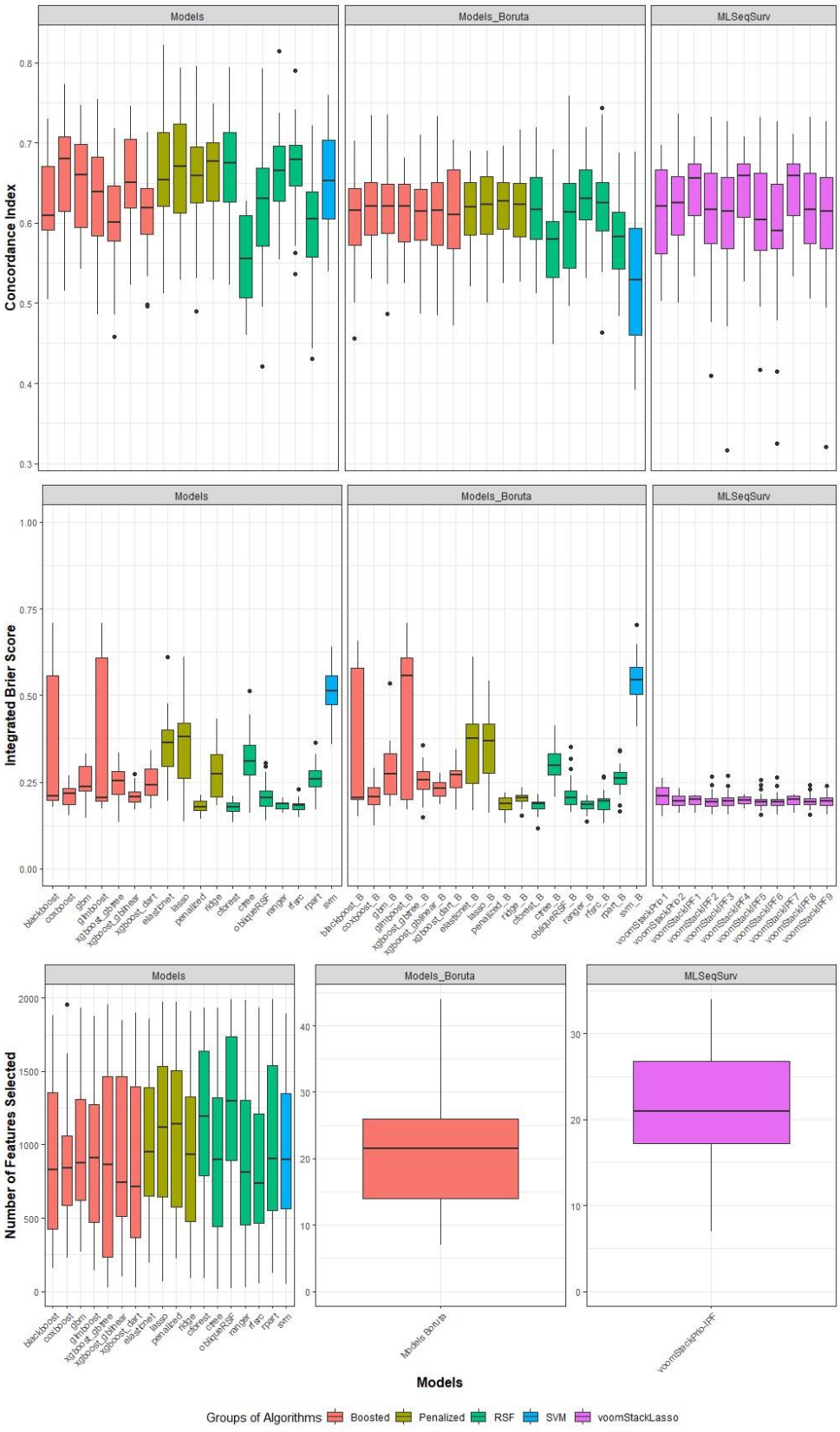


Figure 4.7. The concordance index, integrated Brier score, and the number of selected features for LAML.

Table 4.7. The summary statistics of concordance index, integrated Brier score and the number of features selected for LAML.

Groups of Algorithms	Models	Concordance Index			Integrated Brier Score			The Number of Features Selected		
		Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max
Models	**Blackboost**	0.619±0.065	0.609 (0.589-0.671)	0.505-0.730	0.342±0.195	0.209 (0.196-0.559)	0.177-0.709	889.20±530.61	828.00 (418.25-1385.50)	157-1882
	Cforest	**0.662±0.063**	0.674 (0.624-0.717)	0.523-0.794	**0. 177±0.018**	0.177 (0.165-0.190)	0.134-0.211	1164.33±539.96	1193.00 (746.25-1649.00)	92-1935
	Coxboost	**0.667±0.065**	0.680 (0.610-0.713)	0.515-0.773	0.209±0.032	0.217 (0.184-0.232)	0.152-0.269	884.80±443.94	842.00 (578.00-1121.00)	231-1958
	Ctree	0.554±0.052	0.556 (0.506-0.611)	0.460-0.627	0.314±0.075	0.310 (0.270-0.358)	0.162-0.513	881.20±533.52	896.00 (369.25-1336.00)	13-1930
	Elasticnet	**0.664±0.065**	0.654 (0.619-0.715)	0.513-0.822	0.350±0.095	0.365 (0.286-0.400)	0.196-0.611	1009.83±485.65	952.50 (611.25-1428.75)	192-1858
	Gbm	0.654±0.062	0.661 (0.590-0.700)	0.542-0.747	0.252±0.052	0.237 (0.223-0.303)	0.146-0.333	978.97±478.17	876.50 (616.75-1363.00)	267-1930
	Glmboost	0.630±0.066	0.639 (0.578-0.684)	0.486-0.755	0.376±0.212	0.206 (0.194-0.611)	0.173-0.709	874.33±498.00	908.50 (452.25-1281.00)	140-1875
	Lasso	**0.664±0.070**	0.671 (0.608-0.724)	0.529-0.793	0.355±0.122	0.382 (0.239-0.423)	0.136-0.611	1116.03±531.38	1120.50 (627.50-1552.50)	64-1976
	ObliqueRSF	0.620±0.078	0.630 (0.569-0.671)	0.421-0.792	0.208±0.041	0.204 (0.179-0.227)	0.140-0.306	1256.43±530.96	1295.00 (858.50-1739.25)	21-1992
	Penalized	0.652±0.069	0.660 (0.619-0.696)	0.490-0.796	**0.180±0.019**	0.178 (0.165-0.197)	0.145-0.212	1052.33±533.10	1143.50 (567.25-1517.75)	225-1975
	Ranger	**0.662±0.055**	0.665 (0.626-0.697)	0.555-0.815	**0.183±0.011**	0.187 (0.173-0.190)	0.160-0.206	892.47±564.06	813.50 (398.75-1316.75)	29-1985
	Rfsrc	**0.667±0.057**	0.679 (0.645-0.700)	0.536-0.790	**0.182±0.015**	0.184 (0.171-0.188)	0.148-0.228	872.00±562.09	739.00 (416.75-1275.75)	54-1935
	Ridge	**0.660±0.059**	0.678 (0.618-0.702)	0.529-0.749	0.275±0.073	0.274 (0.205-0.333)	0.184-0.434	922.30±550.44	935.50 (430.00-1347.00)	92-1909
	Rpart	0.592±0.074	0.605 (0.555-0.642)	0.431-0.722	0.260±0.045	0.260 (0.235-0.286)	0.170-0.363	1044.33±586.72	906.50 (535.25-1625.00)	124-1989
	Svm	0.653±0.062	0.653 (0.604-0.706)	0.540-0.760	0.512±0.066	0.513 (0.473-0.557)	0.359-0.639	965.67±501.62	900.00 (542.75-1355.25)	49-1890
	Xgboost (dart)	0.613±0.053	0.619 (0.583-0.646)	0.496-0.713	0.248±0.047	0.240 (0.210-0.289)	0.173-0.342	852.30±613.65	716.50 (338.00-1438.75)	27-1896
	Xgboost (gblinear)	0.648±0.064	0.651 (0.616-0.710)	0.522-0.745	0.208±0.026	0.208 (0.192-0.224)	0.170-0.274	911.23±558.30	741.50 (481.75-1469.25)	99-1845
	Xgboost (gbtree)	0.604±0.059	0.601 (0.577-0.647)	0.458-0.718	0.247±0.046	0.253 (0.210-0.284)	0.133-0.333	927.73±658.62	864.00 (228.75-1500.50)	25-1956
Models Boruta	**Blackboost**	0.606±0.058	0.616 (0.565-0.644)	0.456-0.702	0.367±0.202	0.205 (0.200-0.589)	0.151-0.656	**21.00±8.28**	21.50 (14.00-26.25)	7-44
	Cforest	0.618±0.053	0.617 (0.578-0.658)	0.513-0.719	**0.182±0.021**	0.187 (0.170-0.193)	0.117-0.214			
	Coxboost	0.619±0.051	0.621 (0.583-0.655)	0.530-0.734	0.209±0.035	0.206 (0.181-0.234)	0.125-0.291			
	Ctree	0.571±0.059	0.580 (0.525-0.603)	0.449-0.692	0.301±0.052	0.298 (0.267-0.337)	0.207-0.414			
	Elasticnet	0.616±0.048	0.620 (0.582-0.657)	0.520-0.690	0.353±0.116	0.375 (0.240-0.421)	0.168-0.610			
	Gbm	0.618±0.054	0.621 (0.584-0.650)	0.487-0.735	0.278±0.077	0.273 (0.209-0.339)	0.180-0.534			
	Glmboost	0.617±0.045	0.621 (0.575-0.653)	0.525-0.681	0.435±0.204	0.557 (0.199-0.610)	0.170-0.708			
	Lasso	0.617±0.052	0.623 (0.580-0.661)	0.501-0.690	0.354±0.099	0.370 (0.273-0.425)	0.162-0.543			
	ObliqueRSF	0.603±0.066	0.613 (0.543-0.654)	0.497-0.758	0.215±0.046	0.204 (0.183-0.228)	0.162-0.352			
	Penalized	0.617±0.047	0.627 (0.591-0.654)	0.524-0.696	**0.186±0.023**	0.189 (0.168-0.205)	0.130-0.220			
	Ranger	**0.629±0.048**	0.631 (0.603-0.668)	0.531-0.719	**0.183±0.017**	0.186 (0.171-0.196)	0.135-0.213			
	Rfsrc	**0.622±0.058**	0.626 (0.589-0.651)	0.464-0.743	**0.193±0.030**	0.194 (0.170-0.202)	0.131-0.267			
	Ridge	0.618±0.048	0.623 (0.581-0.653)	0.527-0.716	0.203±0.017	0.205 (0.194-0.214)	0.155-0.234			
	Rpart	0.582±0.057	0.583 (0.538-0.621)	0.484-0.688	0.260±0.038	0.262 (0.242-0.279)	0.166-0.343			
	Svm	0.527±0.085	0.529 (0.457-0.599)	0.392-0.688	0.543±0.070	0.544 (0.500-0.583)	0.411-0.704			
	Xgboost (dart)	0.607±0.067	0.610 (0.565-0.668)	0.472-0.704	0.261±0.043	0.270 (0.228-0.284)	0.171-0.343			
	Xgboost (gblinear)	0.604±0.058	0.616 (0.572-0.652)	0.484-0.733	0.229±0.026	0.231 (0.207-0.252)	0.186-0.276			
	Xgboost (gbtree)	0.613±0.057	0.615 (0.575-0.648)	0.486-0.709	0.252±0.046	0.256 (0.228-0.281)	0.149-0.357			
MLSeqSurv	**voomStackPrio1**	0.615±0.057	0.621 (0.560-0.668)	0.503-0.697	0.209±0.030	0.211 (0.184-0.235)	0.150-0.261	**21.87±7.14**	21.00 (17.00-27.50)	7-34
	voomStackPrio2	0.618±0.058	0.625 (0.577-0.661)	0.500-0.736	**0.196±0.019**	0.196 (0.182-0.212)	0.162-0.231			
	voomStackIPF1	0.638±0.051	0.656 (0.656-0.674)	0.534-0.707	**0.197±0.014**	0.199 (0.185-0.209)	0.162-0.213			
	voomStackIPF2	0.611±0.074	0.617 (0.572-0.666)	0.410-0.732	**0.195±0.022**	0.193 (0.181-0.204)	0.155-0.267			
	voomStackIPF3	0.604±0.085	0.615 (0.565-0.662)	0.317-0.727	**0.196±0.023**	0.194 (0.181-0.205)	0.156-0.269			
	voomStackIPF4	0.639±0.052	0.660 (0.603-0.675)	0.527-0.707	**0.198±0.012**	0.198 (0.186-0.209)	0.174-0.216			
	voomStackIPF5	0.606±0.074	0.604 (0.562-0.666)	0.417-0.732	**0.195±0.021**	0.193 (0.182-0.200)	0.155-0.257			
	voomStackIPF6	0.592±0.086	0.591 (0.565-0.653)	0.324-0.727	**0.196±0.022**	0.193 (0.182-0.202)	0.156-0.263			
	voomStackIPF7	**0.640±0.052**	0.660 (0.604-0.675)	0.534-0.711	**0.197±0.014**	0.199 (0.183-0.209)	0.162-0.214			
	voomStackIPF8	0.617±0.062	0.617 (0.574-0.666)	0.506-0.732	**0.195±0.018**	0.193 (0.183-0.204)	0.155-0.242			
	voomStackIPF9	0.607±0.081	0.615 (0.566-0.662)	0.321-0.727	**0.195±0.020**	0.194 (0.181-0.205)	0.156-0.239			

It was noted that for LAML data, the cforest algorithm, when employed for internal feature selection, achieved the lowest mean integrated Brier score of 0.177. This was followed by penalized (0.180±0.019), rfsrc (0.182±0.015), and ranger(0.183±0.011) algorithms, all with internal feature selection. Meanwhile, the svm algorithm exhibited the highest score (0.512±0.066). In the category of methods from the literature utilizing Boruta feature selection, cforest (cforest_B) (0.182±0.021) and ranger (ranger_B) (0.183±0.017) algorithms showcased the lowest mean integrated Brier score, while svm (svm_B) (0.543±0.070) and glmboost (glmboost_B) (0.435±0.204) algorithms presented the highest mean integrated Brier score. The voomStackIPF2, voomStackIPF5, voomStackIPF8, and voomStackIPF9 algorithms displayed the lowest mean integrated Brier score, at 0.195. Meanwhile, the voomStackPrio1 algorithm (0.209±0.030) displayed the highest mean integrated Brier score.

Among the methods in the literature that applied Boruta feature selection, the mean number of selected features for LAML data was the lowest (21.00±8.28). These were closely followed by the voomStackLasso algorithms (21.87±7.14). In terms of internal feature selection methods, the algorithm with the lowest mean number of features was xgboost with dart (852.30±613.65), while the algorithm with the highest mean number of features was obliqueRSF (1256.43±530.96).

The concordance index, integrated Brier score, and the number of selected features for Brain Lower Grade Glioma (LGG) data are depicted in Figure 4.8, with related summary statistics presented in Table 4.8. After examining both the graph and the table for LGG data, it was observed that coxboost, when applied to internal feature selection, achieved the highest mean concordance index at 0.833. This is followed by the glmboost algorithm (0.832±0.043), with internal feature selection applied, yielding a mean concordance index. The ctree (0.759±0.046) and rpart (0.769±0.038) algorithms exhibited the lowest mean concordance index. In the category of methods from the literature utilizing Boruta feature selection, the glmboost (glmboost_B) (0.832±0.032), elasticnet (elasticnet_B) (0.826±0.035), and lasso (lasso_B) (0.825±0.037) algorithms demonstrated the highest mean concordance index, while the svm(svm_B) algorithm (0.732±0.126) exhibited the lowest mean concordance index. Among the voomStackLasso methods, the voomStackIPF1 (0.817±0.038), voomStackIPF4 (0.816±0.038), and voomStackIPF7 (0.817±0.038) algorithms

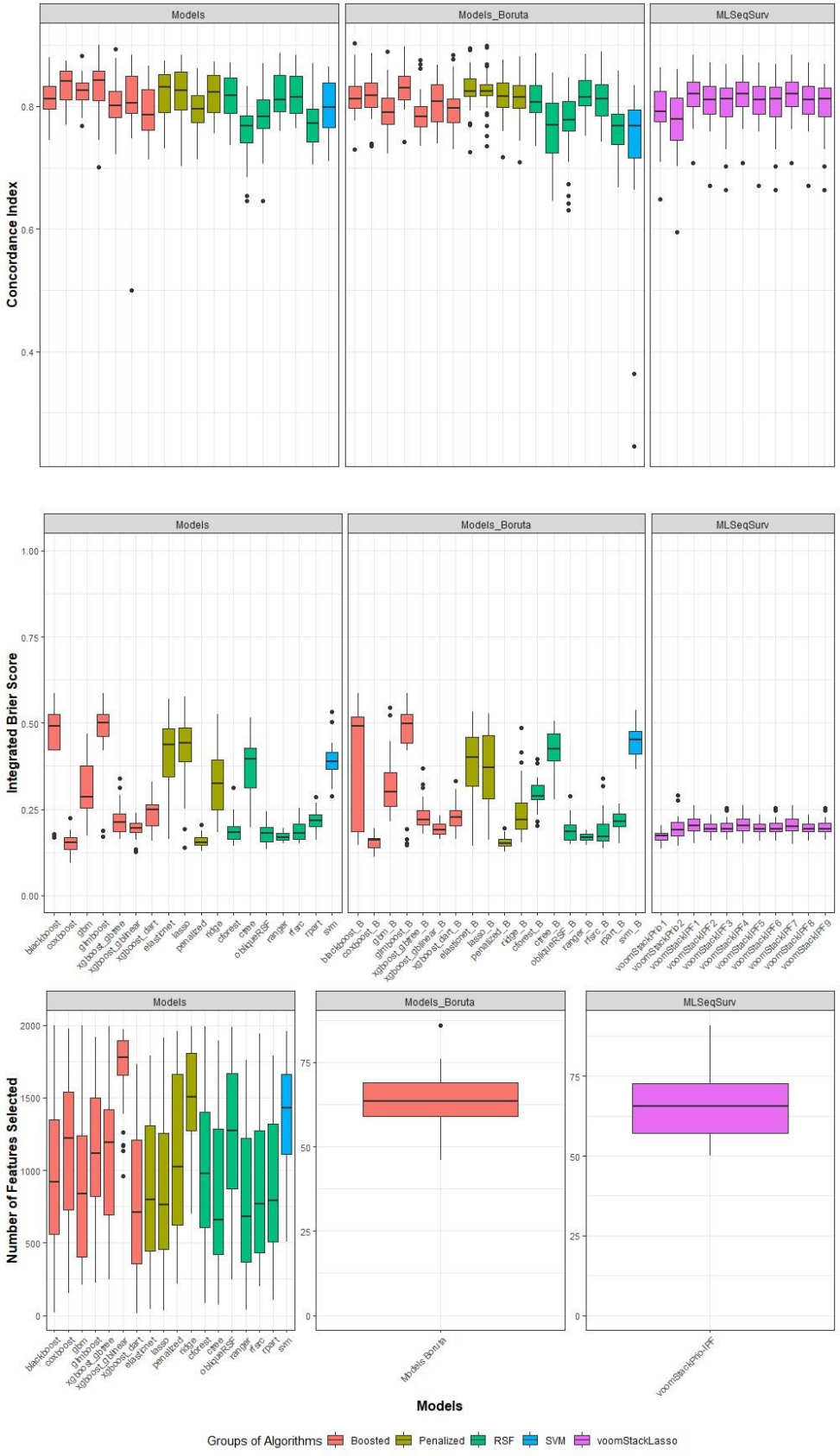


Figure 4.8. The concordance index, integrated Brier score, and the number of selected features for LGG.

Table 4.8. The summary statistics of concordance index, integrated Brier score and the number of features selected for LGG.

Groups of Algorithms	Models	Concordance Index			Integrated Brier Score			The Number of Features Selected		
		Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max
Models	**Blackboost**	0.814±0.032	0.812 (0.793-0.835)	0.745-0.880	0.434±0.139	0.492 (0.422-0.526)	0.167-0.587	984.33±524.46	918.50 (544.00-1381.00)	21-1999
	Cforest	0.817±0.037	0.818 (0.788-0.847)	0.736-0.872	0.189±0.037	0.182 (0.162-0.203)	0.145-0.313	989.97±543.63	975.00 (598.00-1408.50)	87-1992
	Coxboost	**0.833±0.030**	0.841 (0.810-0.859)	0.769-0.874	**0.151±0.028**	0.153 (0.129-0.167)	0.094-0.225	1153.07±525.55	1219.50 (716.75-1548.00)	152-1973
	Ctree	0.759±0.046	0.769 (0.738-0.790)	0.645-0.833	0.377±0.076	0.395 (0.303-0.432)	0.197-0.514	852.27±586.12	655.50 (414.00-1389.00)	71-1892
	Elasticnet	0.823±0.037	0.832 (0.788-0.854)	0.732-0.874	0.411±0.104	0.438 (0.330-0.488)	0.163-0.570	864.73±540.15	799.50 (378.75-1328.25)	41-1790
	Gbm	0.821±0.027	0.826 (0.809-0.839)	0.768-0.882	0.309±0.084	0.286 (0.248-0.378)	0.173-0.469	924.47±559.52	839.50 (371.50-1314.00)	209-1998
	Glmboost	**0.832±0.043**	0.842 (0.808-0.858)	0.701-0.900	0.480±0.091	0.501 (0.455-0.527)	0.171-0.587	1102.50±501.33	1117.00 (739.00-1515.50)	221-1917
	Lasso	0.819±0.045	0.826 (0.792-0.858)	0702-0.884	0.422±0.108	0.443 (0.373-0.490)	0.140-0.577	831.60±522.56	760.50 (419.00-1278.75)	32-1911
	ObliqueRSF	0.783±0.047	0.784 (0.761-0.813)	0.645-0.870	0.180±0.031	0.181 (0.155-0.201)	0.134-0.244	1244.77±484.69	1274.00 (861.00-1678.50)	249-1986
	Penalized	0.796±0.036	0.795 (0.771-0.821)	0.713-0.862	**0.158±0.019**	0.153 (0.145-0.169)	0.128-0.204	1126.27±562.60	1023.50 (593.50-1666.25)	219-1957
	Ranger	0.819±0.033	0.811 (0.791-0.852)	0.760-0.887	0.170±0.012	0.169 (0.161-0.181)	0.151-0.195	815.87±549.94	680.00 (363.25-1253.00)	38-1759
	Rfsrc	0.820±0.035	0.815 (0.787-0.849)	0.764-0.883	0.186±0.026	0.181 (0.162-0.210)	0.152-0.254	889.73±521.20	771.00 (426.00-1313.50)	198-1942
	Ridge	0.819±0.035	0.824 (0.788-0.852)	0.756-0.873	0.329±0.103	0.325 (0.247-0.400)	0.184-0.525	1492.83±335.76	1504.00 (1250.75-1842.75)	698-1991
	Rpart	0.769±0.038	0.772 (0.741-0.798)	0.705-0.869	0.217±0.029	0.216 (0.200-0.235)	0.160-0.286	903.37±484.52	793.50 (475.75-1328.50)	110-1786
	Svm	0.801±0.042	0.798 (0.763-0.839)	0.711-0.865	0.389±0.054	0.387 (0.365-0.419)	0.288-0.533	1367.03±380.29	1432.00 (1087.00-1665.50)	509-1955
	Xgboost (dart)	0.791±0.041	0.786 (0.760-0.832)	0.714-0.866	0.236±0.047	0.248 (0.202-0.267)	0.158-0.330	795.23±535.48	711.50 (350.75-1295.25)	14-1731
	Xgboost (gblinear)	0.805±0.068	0.805 (0.786-0.852)	0.500-0.884	0.194±0.026	0.195 (0.181-0.212)	0.128-0.238	1682.07±285.01	1179.50 (1587.50-1899.25)	962-1968
	Xgboost (gbtree)	0.804±0.036	0.801 (0.781-0.825)	0.722-0.894	0.220±0.046	0.212 (0.183-0.240)	0.164-0.339	1115.03±492.22	1193.50 (681.00-1445.25)	247-1989
Models Boruta	**Blackboost**	0.816±0.034	0.812 (0.796-0.834)	0.729-0.902	0.403±0.159	0.492 (0.182-0.521)	0.147-0.587	**63.53±8.16**	63.50 (58.75-69.25)	46-86
	Cforest	0.813±0.039	0.807 (0.788-0.839)	0.735-0.887	0.298±0.045	0.288 (0.275-0.324)	0.202-0.395			
	Coxboost	0.817±0.034	0.818 (0.798-0.839)	0.735-0.887	**0.155±0.020**	0.160 (0.138-0.169)	0.112-0.196			
	Ctree	0.763±0.053	0.770 (0.723-0.807)	0.645-0.854	0.423±0.057	0.424 (0.389-0.471)	0.278-0.506			
	Elasticnet	**0.826±0.035**	0.824 (0.815-0.849)	0.726-0.894	0.376±0.103	0.400 (0.313-0.463)	0.145-0.533			
	Gbm	0.794±0.040	0.790 (0.770-0.816)	0.722-0.890	0.317±0.086	0.299 (0.255-0.362)	0.216-0.545			
	Glmboost	**0.832±0.032**	0.831 (0.810-0.850)	0.742-0.897	0.453±0.123	0.498 (0.437-0.526)	0.146-0.587			
	Lasso	**0.825±0.037**	0.825 (0.814-0.837)	0.735-0.899	0.362±0.116	0.371 (0.262-0.467)	0.160-0.529			
	ObliqueRSF	0.769±0.058	0.778 (0.757-0.810)	0.631-0.846	0.187±0.032	0.184 (0.160-0.206)	0.148-0.288			
	Penalized	0.817±0.036	0.816 (0.796-0.840)	0.717-0.876	**0.155±0.019**	0.152 (0.142-0.165)	0.126-0.196			
	Ranger	0.820±0.032	0.815 (0.798-0.845)	0.751-0.885	0.168±0.012	0.167 (0.161-0.179)	0.147-0.190			
	Rfsrc	0.810±0.038	0.812 (0.783-0.840)	0.741-0.889	0.189±0.048	0.171 (0.157-0.211)	0.135-0.338			
	Ridge	0.813±0.037	0.815 (0.794-0.835)	0.710-0.880	0.246±0.083	0.220 (0.191-0.275)	0.154-0.485			
	Rpart	0.766±0.037	0.768 (0.738-0.789)	0.668-0.858	0.215±0.027	0.215 (0.199-0.238)	0.150-0.267			
	Svm	0.732±0.126	0.768 (0.715-0.795)	0.246-0.835	0.450±0.048	0.451 (0.410-0.482)	0.366-0.538			
	Xgboost (dart)	0.797±0.037	0.797 (0.770-0.817)	0.730-0.884	0.229±0.041	0.226 (0.201-0.249)	0.163-0.332			
	Xgboost (gblinear)	0.805±0.035	0.808 (0.773-0.838)	0.739-0.867	0.192±0.018	0.189 (0.178-0.209)	0.163-0.231			
	Xgboost (gbtree)	0.789±0.035	0.784 (0.766-0.800)	0.735-0.875	0.230±0.044	0.219 (0.203-0.252)	0.178-0.368			
MLSeqSurv	**voomStackPrio1**	0.793±0.048	0.791 (0.774-0.827)	0.648-0.863	**0.172±0.015**	0.172 (0.160-0.181)	0.136-0.204	**66.40±9.97**	65.50 (57.00-73.25)	50-91
	voomStackPrio2	0.775±0.053	0.779 (0.743-0.815)	0.594-0.861	0.194±0.033	0.189 (0.172-0.211)	0.143-0.290			
	voomStackIPF1	**0.817±0.038**	0.821 (0.797-0.841)	0.708-0.884	0.205±0.026	0.203 (0.186-0.224)	0.150-0.261			
	voomStackIPF2	0.808±0.040	0.811 (0.787-0.836)	0.671-0.871	0.195±0.019	0.193 (0.186-0.208)	0.159-0.234			
	voomStackIPF3	0.802±0.048	0.812 (0.782-0.835)	0.663-0.869	0.197±0.023	0.193 (0.185-0.210)	0.160-0.254			
	voomStackIPF4	**0.816±0.038**	0.820 (0.797-0.841)	0.708-0.884	0.205±0.026	0.203 (0.186-0.224)	0.150-0.261			
	voomStackIPF5	0.808±0.040	0.811 (0.787-0.836)	0.671-0.871	0.195±0.019	0.193 (0.186-0.208)	0.159-0.234			
	voomStackIPF6	0.802±0.049	0.812 (0.782-0.835)	0.663-0.869	0.197±0.023	0.193 (0.185-0.210)	0.159-0.254			
	voomStackIPF7	**0.817±0.038**	0.821 (0.797-0.841)	0.708-0.884	0.205±0.026	0.201 (0.186-0.224)	0.150-0.261			
	voomStackIPF8	0.808±0.040	0.811 (0.787-0.836)	0.671-0.871	0.195±0.019	0.193 (0.186-0.208)	0.159-0.234			
	voomStackIPF9	0.802±0.048	0.812 (0.782-0.835)	0.663-0.869	0.197±0.023	0.193 (0.185-0.210)	0.160-0.254			

showed the highest mean concordance index, while the voomStackPrio2 algorithm (0.775±0.053) displayed the lowest mean concordance index.

It was observed that for LGG data, the coxboost algorithm, when utilized for internal feature selection, attained the lowest mean integrated Brier score of 0.151. This was followed by penalized algorithm (0.158±0.019), with internal feature selection. The glmboost algorithm, with internal feature selection, presented the highest mean integrated Brier score, at 0.480. In the category of methods from the literature utilizing Boruta feature selection, coxboost (coxboost_B) (0.155±0.020) and penalized (penalized_B) (0.155±0.019) algorithms showcased the lowest mean integrated Brier score, while glmboost (glmboost_B) (0.453±0.123) and svm (0.450±0.048) algorithms presented the highest mean integrated Brier score. The voomStackPrio1 algorithm displayed the lowest integrated mean Brier score, at 0.172. Meanwhile, the voomStackIPF1, voomStackIPF4, and voomStackIPF7 algorithms displayed the highest mean integrated Brier score, at 0.205.

Among the methods in the literature that applied Boruta feature selection, the mean number of selected features for LGG data was the lowest (63.53±8.16). These were closely followed by the voomStackLasso algorithms (66.40±9.97). In terms of internal feature selection methods, the algorithm with the lowest mean number of features was xgboost (with booster= “dart”) (795.23±535.48), while the algorithm with the highest mean number of features was xgboost (with booster= “gblinear”) (1682.07±285.01).

The concordance index, integrated Brier score, and the number of selected features for Mesothelioma (MESO) data are depicted in Figure 4.9, with related summary statistics presented in Table 4.9. After examining both the graph and the table for MESO data, it was observed that voomStackIPF1 and voomStackIPF7 algorithms achieved the highest mean concordance index at 0.731. This is followed by the voomStackIPF4 algorithm (0.730±0.066), yielding a mean concordance index. The voomStackPrio2 algorithm exhibited the lowest mean concordance index within the voomStackLasso group, at 0.662±0.065. In the category of methods from the literature utilizing Boruta feature selection, the ridge (ridge_B) (0.729±0.062) and penalized (penalized_B) (0.724±0.060) algorithms demonstrated the highest mean concordance index, while the svm (svm_B) algorithm (0.509±0.185) exhibited the lowest mean concordance index. In the category of methods from the literature utilizing internal

feature selection, the elasticnet (0.728±0.057) and xgboost (with booster= “gblinear”) (0.723±0.058) algorithms demonstrated the highest mean concordance index, while the rpart algorithm(0.609±0.076) exhibited the lowest mean concordance index.

It was observed that the penalized (penalized_B) algorithm, with Boruta feature selection, achieved the lowest mean the integrated Brier score of 0.114 for MESO data. This was followed closely by the cforest (cforest_B) and ranger (ranger_B) algorithms, both employing internal feature selection, with a score of 0.123. The svm (svm_B) algorithm, with the literature employing Boruta feature selection, presented the highest mean integrated Brier score, at 0.624. In the category of methods from the literature utilizing internal feature selection, the penalized algorithm (0.115±0.018) showcased the lowest mean integrated Brier score, while the svm algorithm (0.634±0.058) presented the highest mean integrated Brier score. Among voomStackLasso group, the voomStackIPF6, voomStackIPF8, and voomStackIPF9 algorithms displayed the lowest mean integrated Brier score, at 0.154. The voomStackIPF1, voomStackIPF4, and voomStackIPF7 algorithms displayed the highest mean integrated Brier score, at 0.174.

Among the methods in the literature that applied Boruta feature selection, the mean number of selected features for MESO data was the lowest (26.27±10.03). These were closely followed by the voomStackLasso algorithms (31.57±8.81). In terms of internal feature selection methods, the algorithm with the lowest mean number of features was ranger (637.63±457.30), while the algorithm with the highest mean number of features was obliqueRSF (1195.60±567.02).

The concordance index, integrated Brier score, and the number of selected features for Pancreatic Adenocarcinoma (PAAD) data are depicted in Figure 4.10, with related summary statistics presented in Table 4.10. After examining both the graph and the table for PAAD data, it was observed that voomStackIPF7 and ridge algorithms, with internal feature selection, achieved the highest mean concordance index at 0.640. This is followed by the xgboost (with booster= “gblinear”), with both internal feature selection and Boruta feature selection, yielding a mean concordance index, at 0.639. Among voomStackLasso group, the voomStackIPF7 (0.640±0.052) and voomStackIPF1 (0.637±0.057) algorithms displayed the highest mean concordance index, while voomStackPrio1 algorithm (0.609±0.054) displayed the lowest concordance index. In the category of methods from the literature utilizing Boruta

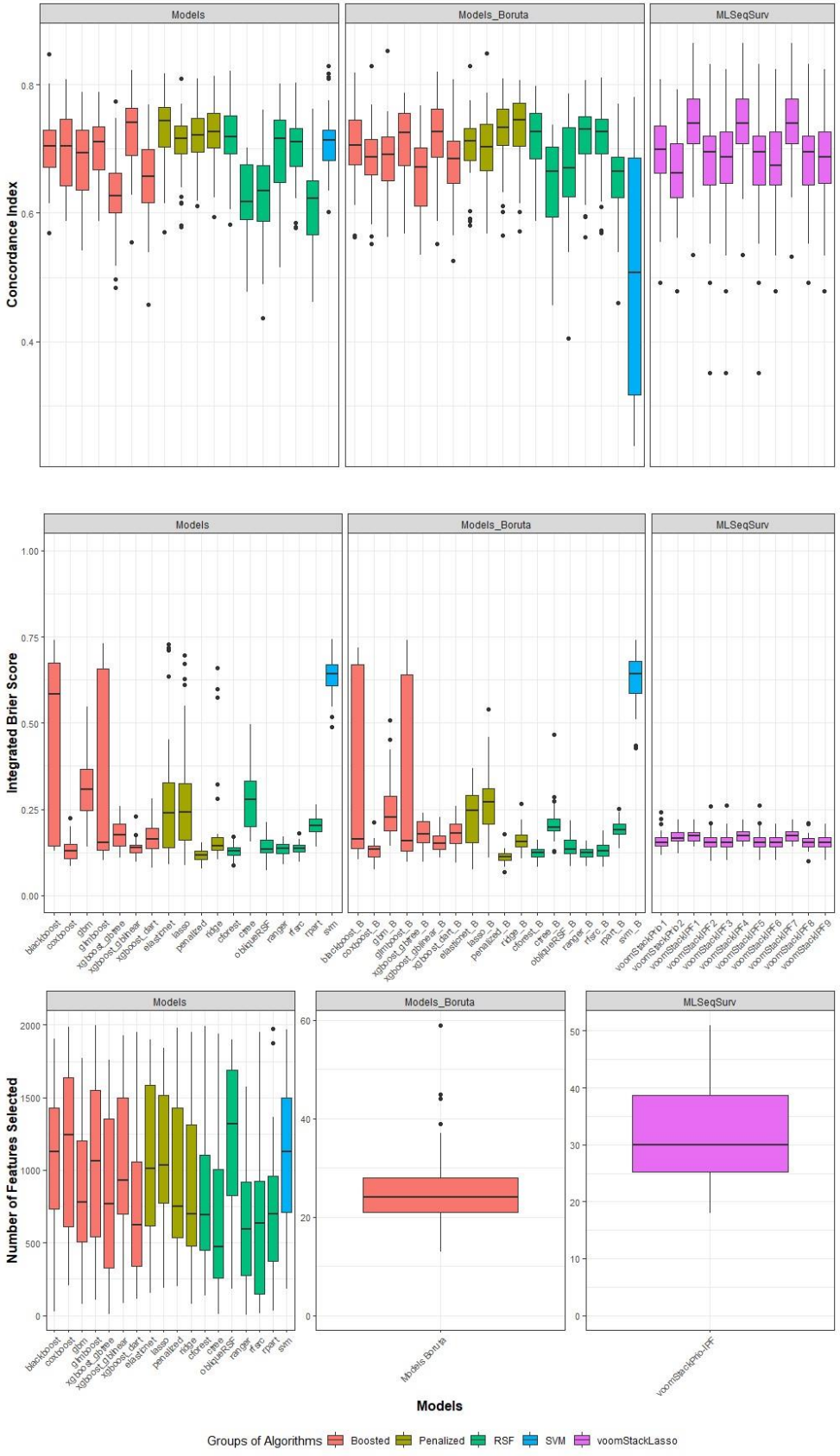


Figure 4.9. The concordance index, integrated Brier score, and the number of selected features for MESO.

Table 4.9. The summary statistics of concordance index, integrated Brier score and the number of features selected for MESO.

Groups of Algorithms	Models	Concordance Index			Integrated Brier Score			The Number of Features Selected		
		Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max
Models	**Blackboost**	0.704±0.061	0.704 (0.669-0.732)	0.569-0.848	0.440±0.267	0.583 (0.142-0.676)	0.128-0.741	1084.43±529.90	1129.50 (692.75-1465.25)	27-1906
	Cforest	0.712±0.061	0.718 (0.683-0.753)	0.582-0.822	0.129±0.019	0.129 (0.117-0.140)	0.087-0.172	843.67±510.99	693.00 (438.75-1196.25)	135-1990
	Coxboost	0.699±0.068	0.704 (0.636-0.749)	0.587-0.809	0.134±0.035	0.130 (0.107-0.155)	0.085-0.226	1163.30±574.48	1246.00 (583.75-1678.50)	204-1983
	Ctree	0.620±0.059	0.618 (0.589-0.682)	0.478-0.702	0.282±0.091	0.279 (0.192-0.335)	0.157-0.497	699.40±635.02	474.50 (242.00-1169.25)	11-1939
	Elasticnet	**0.728±0.057**	0.743 (0.696-0.768)	0.571-0.817	0.287±0.190	0.238 (0.135-0.343)	0.090-0.729	1073.77±553.66	1010.50 (599.50-1598.50)	153-1900
	Gbm	0.684±0.064	0.694 (0.627-0.731)	0.542-0.788	0.308±0.088	0.307 (0.239-0.371)	0.142-0.546	852.40±500.67	780.50 (463.25-1255.25)	78-1771
	Glmboost	0.702±0.051	0.711 (0.660-0.735)	0.588-0.788	0.337±0.267	0.153 (0.131-0.660)	0.101-0.730	1045.40±567.91	1063.00 (494.25-1567.25)	109-1998
	Lasso	0.702±0.056	0.716 (0.683-0.737)	0.579-0.810	0.297±0.179	0.241 (0.159-0.369)	0.087-0.697	1075.67±505.07	1032.00 (740.00-1570.50)	186-1838
	ObliqueRSF	0.622±0.082	0.634 (0.583-0.676)	0.436-0.761	0.137±0.030	0.133 (0.124-0.162)	0.073-0.213	1195.60±567.02	1321.00 (810.50-1715.50)	184-1899
	Penalized	0.713±0.055	0.721 (0.685-0.751)	0.611-0.810	**0.115±0.018**	0.118 (0.103-0.130)	0.077-0.153	910.47±526.21	749.00 (517.00-1443.75)	202-1977
	Ranger	0.700±0.064	0.716 (0.641-0.746)	0.515-0.802	0.134±0.019	0.137 (0.120-0.149)	0.091-0.170	637.63±457.30	594.50 (245.50-919.25)	5-1572
	Rfsrc	0.699±0.060	0.710 (0.668-0.736)	0.577-0.803	0.136±0.018	0.136 (0.125-0.147)	0.098-0.181	647.37±553.53	637.50 (143.75-934.50)	17-1951
	Ridge	0.718±0.053	0.726 (0.695-0.756)	0.594-0.813	0.198±0.147	0.143 (0.131-0.171)	0.105-0.659	881.00±536.39	698.50 (466.75-1350.50)	78-1951
	Rpart	0.609±0.076	0.623 (0.554-0.651)	0.462-0.761	0.200±0.030	0.202 (0.183-0.223)	0.141-0.263	750.03±488.10	700.00 (360.25-1021.00)	34-1971
	Svm	0.715±0.053	0.714 (0.680-0.732)	0.602-0.830	0.634±0.058	0.644 (0.605-0.673)	0.490-0.743	1080.63±503.17	1125.00 (684.75-1524.50)	185-1967
	Xgboost (dart)	0.657±0.065	0.657 (0.612-0.700)	0.457-0.769	0.167±0.046	0.162 (0.133-0.197)	0.079-0.280	715.23±470.47	622.00 (327.75-1120.75)	116-1948
	Xgboost (gblinear)	**0.723±0.058**	0.742 (0.685-0.766)	0.555-0.823	0.139±0.026	0.139 (0.124-0.147)	0.096-0.230	1094.50±517.51	933.00 (688.00-1538.50)	87-1926
	Xgboost (gbtree)	0.629±0.073	0.626 (0.596-0.668)	0.483-0.774	0.177±0.042	0.176 (0.141-0.211)	0.109-0.258	820.23±550.03	769.00 (318.75-1409.25)	8-1761
Models Boruta	**Blackboost**	0.697±0.065	0.706 (0.668-0.749)	0.562-0.818	0.385±0.268	0.164 (0.137-0.673)	0.106-0.719	**26.27±10.03**	24.00 (20.50-28.25)	13-59
	Cforest	0.712±0.060	0.726 (0.678-0.758)	0.588-0.797	**0.123±0.017**	0.124 (0.110-0.135)	0.082-0.160			
	Coxboost	0.684±0.063	0.688 (0.652-0.717)	0.551-0.830	0.131±0.025	0.134 (0.111-0.145)	0.074-0.212			
	Ctree	0.648±0.072	0.666 (0.590-0.705)	0.457-0.737	0.211±0.060	0.198 (0.186-0.223)	0.127-0.465			
	Elasticnet	0.705±0.058	0.712 (0.681-0.736)	0.581-0.830	0.226±0.084	0.246 (0.149-0.294)	0.076-0.369			
	Gbm	0.683±0.062	0.691 (0.645-0.724)	0.562-0.852	0.251±0.093	0.228 (0.185-0.297)	0.143-0.508			
	Glmboost	0.706±0.059	0.725 (0.671-0.756)	0.568-0.787	0.316±0.259	0.157 (0.129-0.657)	0.098-0.741			
	Lasso	0.699±0.061	0.702 (0.661-0.739)	0.568-0.849	0.268±0.098	0.270 (0.198-0.313)	0.110-0.540			
	ObliqueRSF	0.660±0.086	0.670 (0.611-0.734)	0.404-0.785	0.140±0.032	0.134 (0.120-0.163)	0.085-0.218			
	Penalized	**0.724±0.060**	0.733 (0.699-0.765)	0.565-0.810	**0.114±0.020**	0.113 (0.100-0.125)	0.069-0.178			
	Ranger	0.713±0.061	0.731 (0.690-0.751)	0.562-0.807	**0.123±0.016**	0.124 (0.110-0.135)	0.086-0.158			
	Rfsrc	0.711±0.065	0.727 (0.691-0.746)	0.569-0.810	0.133±0.026	0.130 (0.113-0.147)	0.083-0.189			
	Ridge	**0.729±0.062**	0.745 (0.699-0.772)	0.572-0.807	0.162±0.033	0.156 (0.142-0.176)	0.108-0.266			
	Rpart	0.654±0.068	0.666 (0.621-0.691)	0.460-0.770	0.191±0.026	0.191 (0.178-0.208)	0.137-0.252			
	Svm	0.509±0.185	0.508 (0.312-0.689)	0.237-0.780	0.624±0.079	0.642 (0.585-0.683)	0.428-0.742			
	Xgboost (dart)	0.673±0.067	0.684 (0.638-0.713)	0.525-0.808	0.179±0.037	0.181 (0.149-0.208)	0.096-0.258			
	Xgboost (gblinear)	0.717±0.065	0.727 (0.684-0.764)	0.552-0.820	0.157±0.030	0.152 (0.132-0.174)	0.111-0.228			
	Xgboost (gbtree)	0.660±0.068	0.671 (0.603-0.705)	0.535-0.767	0.181±0.037	0.177 (0.152-0.216)	0.098-0.240			
MLSeqSurv	**voomStackPrio1**	0.691±0.072	0.699 (0.660-0.743)	0.492-0.808	0.160±0.028	0.154 (0.143-0.169)	0.118-0.241	**31.57±8.81**	30.00 (24.75-39.00)	18-51
	voomStackPrio2	0.662±0.065	0.662 (0.622-0.712)	0.478-0.792	0.170±0.023	0.165 (0.156-0.183)	0.122-0.220			
	voomStackIPF1	**0.731±0.066**	0.740 (0.705-0.782)	0.535-0.864	0.174±0.020	0.173 (0.159-0.185)	0.141-0.220			
	voomStackIPF2	0.669±0.092	0.695 (0.641-0.723)	0.351-0.832	0.157±0.029	0.154 (0.139-0.168)	0.101-0.258			
	voomStackIPF3	0.668±0.093	0.687 (0.644-0.729)	0.351-0.824	0.157±0.029	0.154 (0.139-0.168)	0.103-0.262			
	voomStackIPF4	**0.730±0.066**	0.740 (0.705-0.781)	0.535-0.864	0.174±0.020	0.173 (0.159-0.186)	0.142-0.220			
	voomStackIPF5	0.669±0.092	0.695 (0.641-0.723)	0.351-0.832	0.157±0.029	0.154 (0.139-0.167)	0.102-0.261			
	voomStackIPF6	0.675±0.072	0.674 (0.642-0.729)	0.478-0.824	**0.154±0.022**	0.154 (0.139-0.168)	0.103-0.207			
	voomStackIPF7	**0.731±0.067**	0.740 (0.705-0.782)	0.532-0.864	0.174±0.020	0.173 (0.159-0.185)	0.141-0.220			
	voomStackIPF8	0.678±0.070	0.695 (0.642-0.723)	0.492-0.832	**0.154±0.022**	0.154 (0.139-0.167)	0.101-0.209			
	voomStackIPF9	0.677±0.072	0.687 (0.644-0.729)	0.478-0.824	**0.154±0.022**	0.154 (0.139-0.168)	0.103-0.207			

feature selection, the xgboost (xgboost_gblinear_B) (with booster= “gblinear”) algorithm (0.639±0.032) demonstrated the highest mean concordance index, while the svm (svm_B) algorithm (0.528±0.110) exhibited the lowest mean concordance index. In the group of methods from the literature employing internal feature selection, the ridge (0.640±0.050) and xgboost (with booster= “gblinear”) (0.639±0.040) algorithms demonstrated the highest mean concordance index, while the ctree algorithm (0.553±0.050) exhibited the lowest mean concordance index.

It was observed that the penalized algorithm, with both internal feature selection and Boruta feature selection, achieved the lowest mean integrated Brier score at 0.162 for PAAD data. In the category of methods from the literature utilizing internal feature selection, the penalized (0.162±0.021) and cforest (0.166±0.019) algorithms showcased the lowest mean integrated Brier score, while the svm (0.520±0.057), blackboost (0.505±0.139) and ctree (0.419±0.076) algorithms presented the highest mean integrated Brier score. In the category of methods from the literature employing Boruta feature selection, the penalized (penalized_B) (0.162±0.024), coxboost (coxboost_B) (0.176±0.035), and ranger (ranger_B) (0.177±0.020) algorithms showcased the lowest mean integrated Brier score, while svm (svm_B) (0.540±0.062), glmboost (glmboost_B) (0.477±0.158), and blackboost (blackboost_B) (0.462±0.172) algorithms presented the highest mean integrated Brier score. Among voomStackLasso group, the voomStackPrio2 (0.172±0.021) and voomStackIPF5 (0.175±0.017) algorithms displayed the lowest mean integrated Brier score. Conversely, the voomStackPrio1 algorithm displayed the highest mean integrated Brier score, at 0.187.

Among the voomStackLasso algorithms, the mean number of selected features for PAAD data was the lowest (8.70±3.25). These were closely followed by the methods in the literature that utilized Boruta feature selection (9.20±3.39). In terms of internal feature selection methods, the algorithm with the lowest mean number of features was rfsrc (578.70±465.45), while the algorithm with the highest mean number of features was obliqueRSF (1062.10±515.55).

The concordance index, integrated Brier score, and the number of selected features for Sarcoma (SARC) data are depicted in Figure 4.11, with related summary statistics presented in Table 4.11. After examining both the graph and the table for SARC data, it was observed that the ridge algorithm, when applied for internal feature

selection, achieved the highest mean concordance index at 0.650. In the group of methods from the literature employing internal feature selection, the ridge (0.650±0.042), cforest (0.635±0.042), penalized (0.634±0.041) and rfsrc (0.634±0.032) algorithms demonstrated the highest mean concordance index, while the rpart algorithm (0.557±0.054) exhibited the lowest mean concordance index. In the group of methods from the literature employing Boruta feature selection, the ranger (ranger_B) algorithm (0.619±0.052) demonstrated the highest mean concordance index, while the svm (svm_B) algorithm (0.553±0.055) exhibited the lowest mean concordance index. Among voomStackLasso group, the voomStackIPF7 algorithm (0.615±0.032) displayed the highest mean concordance index, while voomStackPrio2 algorithm (0.597±0.043) displayed the lowest mean concordance index.

It was observed that for SARC data, the voomStackIPF4 and voomStackIPF7 algorithms achieved the lowest mean integrated Brier score of 0.192. This is followed by voomStackIPF1, voomStackIPF2, and voomStackIPF8 resulting in an integrated Brier score of 0.193. In the category of methods from the literature utilizing internal feature selection, the cforest and ranger algorithms showcased the lowest mean integrated Brier score at 0.206, while the svm algorithm (0.437±0.062) presented the highest mean integrated Brier score. In the category of methods from the literature utilizing Boruta feature selection, ridge (ridge_B) (0.207±0.008) and ranger (0.209±0.017) algorithms showcased the lowest mean integrated Brier score, while svm (svm_B) (0.450±0.062) and blackboost (blackboost_B) (0.418±0.111) algorithms presented the highest mean integrated Brier score. Among voomStackLasso group, the voomStackIPF4 (0.192±0.010), voomStackIPF7 (0.192±0.011), voomStackIPF1 (0.193±0.011), voomStackIPF2 (0.193±0.012), and voomStackIPF8 (0.193±0.013) algorithms displayed the lowest mean integrated Brier score. Conversely, the voomStackPrio1 algorithm displayed the highest mean integrated Brier score, at 0.202.

Among the methods in the literature that applied Boruta feature selection, the mean number of selected features for SARC data was the lowest (14.87±4.33). These were closely followed by the voomStackLasso algorithms (16.27±4.81). In terms of internal feature selection methods, the algorithm with the lowest mean number of features was ranger (703.37±511.13), while the algorithm with the highest mean number of features was xgboost (with booster= “gblinear”) (1184.37±519.21).

The concordance index, integrated Brier score, and the number of selected features for Uveal Melanoma (UVM) data are depicted in Figure 4.12, with related summary statistics presented in Table 4.12. After examining both the graph and the table for UVM data, it was observed that voomStackIPF3 and voomStackIPF9 achieved the highest mean concordance index at 0.841. In the group of methods from the literature employing internal feature selection, the elasticnet (0.819±0.056), xgboost (with booster= "gblinear") (0.816±0.056), and glmboost (0.815±0.068) algorithms demonstrated the highest mean concordance index, while the rpart algorithm (0.698±0.106) exhibited the lowest mean concordance index. In the group of methods from the literature employing Boruta feature selection, the xgboost (with booster= "gblinear") (xgboost_gblinear_B) algorithm (0.839±0.061) demonstrated the highest mean concordance index, while the rpart (rpart_B) algorithm (0.721±0.090) exhibited the lowest mean concordance index. Among voomStackLasso group, the voomStackIPF3 and voomStackIPF9 algorithm (0.841±0.059) displayed the highest mean concordance index while voomStackIPF1, voomStackIPF4, and voomStackIPF7 algorithms displayed the lowest mean concordance index, at 0.817.

It was observed that for UVM data, the voomStackIPF3 and voomStackIPF9 algorithms achieved the lowest mean integrated Brier score of 0.108. This is followed voomStackIPF2, voomStackIPF5, voomStackIPF6, and voomStackIPF8 yielding an integrated Brier score of 0.111. The voomStackIPF1, voomStackIPF4, and voomStackIPF7 algorithms displayed the highest mean integrated Brier score, at 0.144. In the category of methods from the literature utilizing internal feature selection, the penalized algorithm showcased the lowest mean integrated Brier score at 0.122, while the svm algorithm (0.303±0.071) presented the highest mean integrated Brier score. In the category of methods from the literature employing Boruta feature selection, penalized (penalized_B) algorithm (0.119±0.019) showcased the lowest mean integrated Brier score, while gbm algorithm (0.312±0.086) presented the highest mean integrated Brier score.

Among the voomStackLasso algorithms, the mean number of selected features for UVM data was the lowest (42.20±12.95). These were closely followed by the methods in the literature that utilized Boruta feature selection (43.57±12.03). In terms of internal feature selection methods, the algorithm with the lowest mean number of

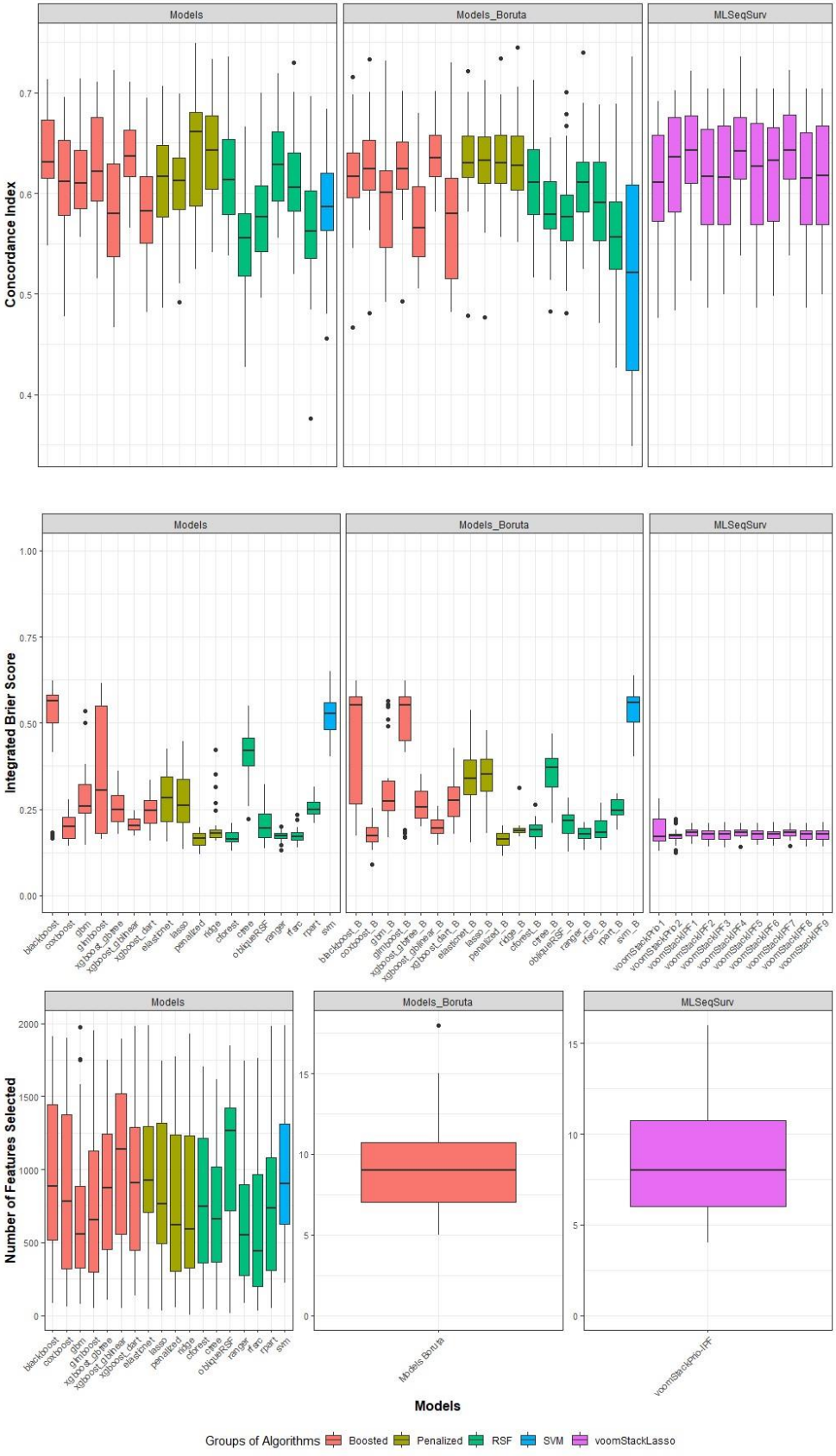


Figure 4.10. The concordance index, integrated Brier score, and the number of selected features for PAAD.

Table 4.10. The summary statistics of concordance index, integrated Brier score and the number of features selected for PAAD.

Groups of Algorithms	Models	Concordance Index			Integrated Brier Score			The Number of Features Selected		
		Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max
Models	**Blackboost**	**0.637±0.044**	0.632 (0.613-0.675)	0.548-0.713	0.505±0.139	0.563 (0.485-0.582)	0.165-0.622	963.83±583.64	883.00 (486.50-1549.50)	82-1913
	Cforest	0.618±0.050	0.613 (0.578-0.658)	0.538-0.736	**0.166±0.019**	0.164 (0.154-0.184)	0.129-0.210	800.27±523.94	746.50 (319.75-1266.50)	42-1703
	Coxboost	0.611±0.056	0.612 (0.575-0.655)	0.478-0.696	0.201±0.039	0.200 (0.165-0.228)	0.144-0.277	860.90±588.22	782.00 (291.00-1436.25)	58-1898
	Ctree	0.553±0.050	0.556 (0.516-0.580)	0.427-0.666	0.419±0.076	0.420 (0.372-0.464)	0.221-0.551	717.90±443.08	663.00 (355.25-1032.00)	38-1620
	Elasticnet	0.613±0.050	0.617 (0.576-0.650)	0.486-0.707	0.283±0.077	0.283 (0.214-0.348)	0.156-0.424	985.90±455.34	925.50 (682.75-1315.75)	41-1986
	Gbm	0.616±0.041	0.610 (0.584-0.644)	0.556-0.714	0.282±0.088	0.259 (0.235-0.327)	0.146-0.536	705.63±541.56	554.50 (320.75-930.25)	79-1973
	Glmboost	0.627±0.053	0.622 (0.591-0.679)	0.515-0.711	0.360±0.187	0.304 (0.180-0.550)	0.163-0.615	772.63±586.42	652.00 (280.75-1229.25)	49-1952
	Lasso	0.609±0.049	0.613 (0.581-0.639)	0.492-0.699	0.275±0.084	0.261 (0.205-0.343)	0.133-0.446	883.93±530.35	764.50 (480.50-1334.50)	30-1743
	ObliqueRSF	0.577±0.050	0.577 (0.540-0.608)	0.496-0.700	0.205±0.052	0.194 (0.167-0.241)	0.136-0.322	1062.10±515.55	1267.00 (688.50-1448.50)	14-1849
	Penalized	**0.636±0.062**	0.661 (0.581-0.682)	0.525-0.749	**0.162±0.021**	0.166 (0.146-0.181)	0.120-0.198	774.13±549.19	621.00 (294.50-1275.75)	54-1776
	Ranger	0.631±0.049	0.629 (0.588-0.663)	0.555-0.719	0.172±0.015	0.173 (0.165-0.181)	0.132-0.200	673.80±448.33	553.50 (262.50-923.75)	83-1742
	Rfsrc	0.611±0.048	0.606 (0.580-0.643)	0.519-0.730	0.172±0.021	0.171 (0.160-0.184)	0.139-0.234	578.70±465.45	442.00 (182.75-990.00)	34-1764
	Ridge	**0.640±0.050**	0.643 (0.601-0.681)	0.541-0.733	0.201±0.061	0.181 (0.168-0.190)	0.158-0.424	776.73±583.27	591.00 (313.50-1258.75)	2-1929
	Rpart	0.566±0.060	0.563 (0.533-0.603)	0.376-0.697	0.255±0.027	0.249 (0.236-0.275)	0.211-0.314	760.67±533.97	734.50 (295.00-1136.50)	51-1982
	Svm	0.588±0.055	0.587 (0.560-0.622)	0.456-0.684	0.520±0.057	0.528 (0.480-0.561)	0.402-0.649	988.83±496.60	901.50 (603.00-1330.75)	224-1988
	Xgboost (dart)	0.587±0.049	0.582 (0.551-0.621)	0.482-0.694	0.245±0.042	0.245 (0.208-0.280)	0.159-0.336	918.40±548.36	908.50 (401.25-1299.50)	137-1981
	Xgboost (gblinear)	**0.639±0.040**	0.637 (0.613-0.666)	0.566-0.710	0.206±0.021	0.202 (0.190-0.222)	0.173-0.247	1037.30±586.67	1140.50 (529.00-1540.50)	52-1897
	Xgboost (gbtree)	0.583±0.059	0.580 (0.531-0.631)	0.467-0.722	0.252±0.050	0.249 (0.213-0.293)	0.177-0.361	842.60±500.42	872.50 (439.75-1257.25)	107-1751
Models Boruta	**Blackboost**	0.614±0.051	0.617 (0.588-0.642)	0.467-0.716	0.462±0.172	0.552 (0.198-0.579)	0.174-0.622	**9.20±3.39**	9.00 (6.75-11.00)	5-18
	Cforest	0.613±0.048	0.611 (0.576-0.646)	0.516-0.712	0.189±0.027	0.191 (0.168-0.206)	0.135-0.263			
	Coxboost	0.626±0.046	0.624 (0.602-0.656)	0.481-0.733	**0.176±0.035**	0.174 (0.155-0.202)	0.089-0.254			
	Ctree	0.582±0.037	0.579 (0.564-0.613)	0.483-0.655	0.358±0.057	0.371 (0.315-0.398)	0.211-0.468			
	Elasticnet	**0.632±0.044**	0.630 (0.614-0.659)	0.479-0.721	0.340±0.083	0.339 (0.292-0.396)	0.154-0.538			
	Gbm	0.593±0.056	0.601 (0.541-0.624)	0.492-0.732	0.310±0.112	0.273 (0.244-0.336)	0.168-0.564			
	Glmboost	0.626±0.041	0.624 (0.603-0.653)	0.493-0.702	0.477±0.158	0.552 (0.437-0.577)	0.169-0.622			
	Lasso	**0.631±0.045**	0.633 (0.608-0.657)	0.477-0.712	0.338±0.081	0.351 (0.285-0.396)	0.179-0.478			
	ObliqueRSF	0.586±0.052	0.576 (0.551-0.608)	0.481-0.701	0.209±0.041	0.217 (0.179-0.238)	0.126-0.282			
	Penalized	**0.633±0.039**	0.630 (0.608-0.662)	0.556-0.735	**0.162±0.024**	0.163 (0.145-0.182)	0.114-0.203			
	Ranger	0.610±0.047	0.611 (0.581-0.633)	0.525-0.740	**0.177±0.020**	0.178 (0.165-0.196)	0.131-0.212			
	Rfsrc	0.589±0.056	0.591 (0.550-0.635)	0.471-0.688	0.192±0.035	0.183 (0.168-0.217)	0.131-0.267			
	Ridge	0.630±0.042	0.628 (0.598-0.659)	0.552-0.745	0.192±0.024	0.188 (0.184-0.194)	0.170-0.313			
	Rpart	0.561±0.061	0.557 (0.520-0.593)	0.427-0.689	0.252±0.026	0.247 (0.232-0.280)	0.189-0.297			
	Svm	0.528±0.110	0.521 (0.422-0.611)	0.349-0.736	0.540±0.062	0.560 (0.496-0.577)	0.402-0.639			
	Xgboost (dart)	0.571±0.061	0.580 (0.513-0.618)	0.482-0.730	0.274±0.060	0.275 (0.226-0.315)	0.179-0.427			
	Xgboost (gblinear)	**0.639±0.032**	0.635 (0.616-0.662)	0.581-0.702	0.197±0.028	0.195 (0.178-0.220)	0.147-0.259			
	Xgboost (gbtree)	0.572±0.041	0.566 (0.534-0.609)	0.505-0.680	0.264±0.049	0.255 (0.222-0.311)	0.199-0.352			
MLSeqSurv	**voomStackPrio1**	0.609±0.054	0.611 (0.570-0.660)	0.476-0.692	0.187±0.042	0.171 (0.158-0.222)	0.128-0.280	**8.70±3.25**	8.00 (6.00-11.25)	4-16
	voomStackPrio2	0.620±0.064	0.636 (0.578-0.676)	0.484-0.702	**0.172±0.021**	0.172 (0.164-0.180)	0.125-0.223			
	voomStackIPF1	**0.637±0.057**	0.643 (0.607-0.680)	0.513-0.722	0.182±0.014	0.183 (0.172-0.191)	0.150-0.210			
	voomStackIPF2	0.615±0.058	0.617 (0.567-0.666)	0.486-0.704	**0.177±0.019**	0.178 (0.162-0.188)	0.140-0.211			
	voomStackIPF3	0.614±0.058	0.616 (0.566-0.669)	0.499-0.704	**0.177±0.019**	0.178 (0.162-0.188)	0.140-0.212			
	voomStackIPF4	0.641±0.054	0.642 (0.613-0.678)	0.538-0.736	0.182±0.014	0.183 (0.173-0.192)	0.142-0.210			
	voomStackIPF5	0.617±0.061	0.627 (0.567-0.673)	0.486-0.704	**0.175±0.017**	0.179 (0.161-0.187)	0.145-0.211			
	voomStackIPF6	0.616±0.062	0.633 (0.569-0.666)	0.498-0.704	**0.177±0.017**	0.179 (0.165-0.186)	0.144-0.212			
	voomStackIPF7	**0.640±0.052**	0.643 (0.613-0.680)	0.538-0.722	0.181±0.015	0.183 (0.172-0.191)	0.144-0.209			
	voomStackIPF8	0.614±0.058	0.615 (0.567-0.664)	0.486-0.704	**0.177±0.019**	0.178 (0.162-0.189)	0.140-0.211			
	voomStackIPF9	0.615±0.058	0.617 (0.566-0.669)	0.499-0.704	**0.177±0.019**	0.178 (0.162-0.188)	0.141-0.212			

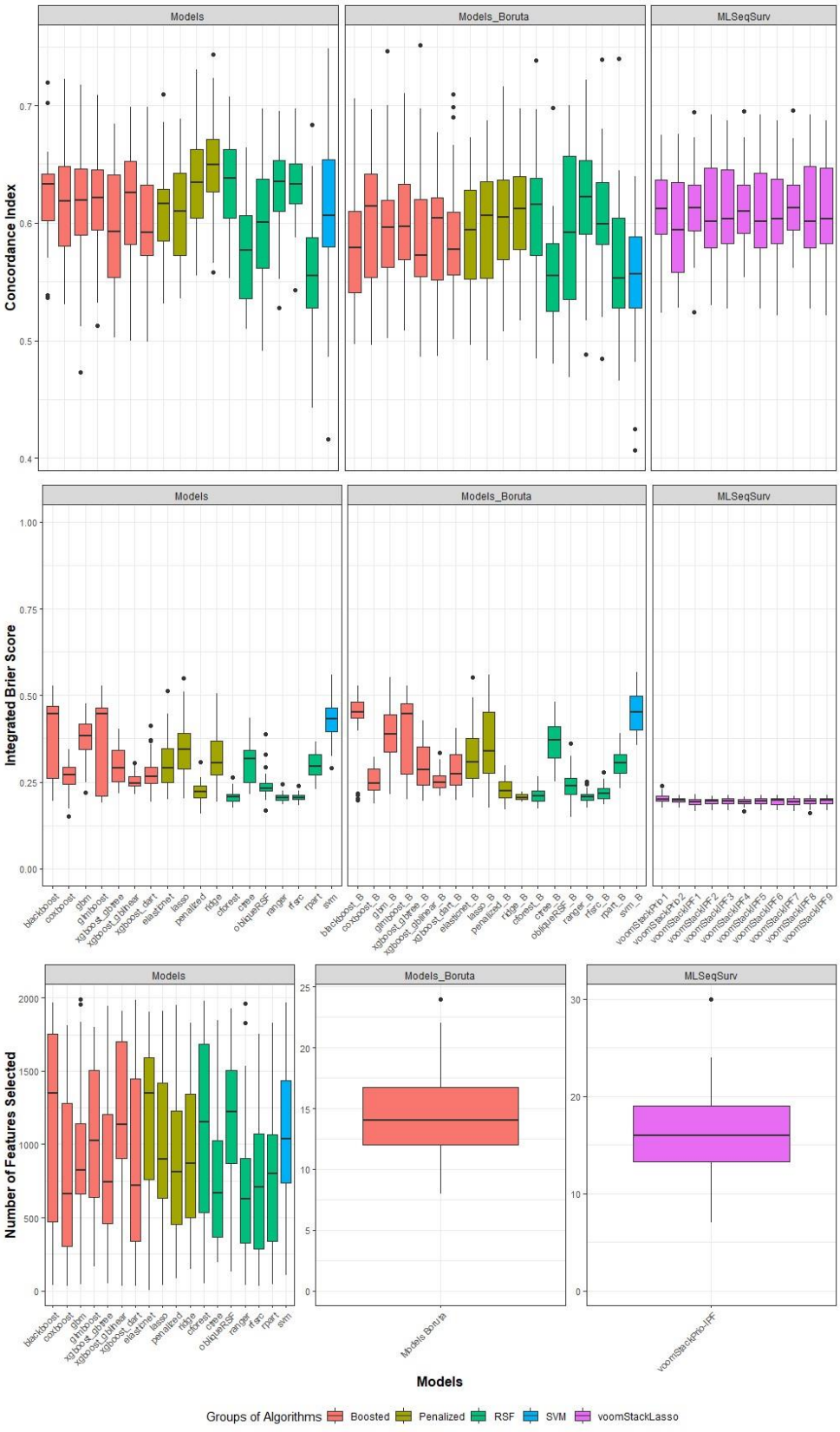


Figure 4.11. The concordance index, integrated Brier score, and the number of selected features for SARC.

Table 4.11. The summary statistics of concordance index, integrated Brier score and the number of features selected for SARC.

Groups of Algorithms	Models	Concordance Index			Integrated Brier Score			The Number of Features Selected		
		Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max
Models	**Blackboost**	0.624±0.039	0.633 (0.601-0.643)	0.536-0.719	0.395±0.121	0.448 (0.215-0.473)	0.195-0.527	1136.70±681.49	1352.00 (443.00-1798.75)	37-1965
	Cforest	**0.635±0.042**	0.638 (0.603-0.666)	0.553-0.707	**0.206±0.018**	0.207 (0.193-0.214)	0.176-0.263	1065.40±630.42	1151.00 (496.00-1696.00)	48-1978
	Coxboost	0.617±0.050	0.619 (0.577-0.649)	0.531-0.723	0.264±0.046	0.271 (0.239-0.294)	0.151-0.345	824.57±607.83	661.00 (286.25-1373.50)	34-1811
	Ctree	0.574±0.044	0.577 (0.534-0.608)	0.510-0.664	0.308±0.066	0.319 (0.242-0.351)	0.214-0.435	766.70±486.35	670.00 (352.00-1053.00)	195-1848
	Elasticnet	0.611±0.040	0.616 (0.583-0.629)	0.531-0.709	0.305±0.075	0.290 (0.247-0.349)	0.200-0.513	1138.87±585.43	1349.00 (682.50-1634.50)	4-1904
	Gbm	0.614±0.052	0.619 (0.587-0.647)	0.473-0.717	0.378±0.066	0.383 (0.342-0.425)	0.218-0.476	892.43±507.18	825.00 (606.00-1168.00)	43-1992
	Glmboost	0.617±0.046	0.621 (0.592-0.647)	0.512-0.709	0.371±0.130	0.446 (0.209-0.468)	0.189-0.527	1016.47±519.40	1027.00 (601.25-1518.00)	164-1798
	Lasso	0.609±0.044	0.610 (0.571-0.644)	0.536-0.689	0.343±0.085	0.344 (0.281-0.397)	0.203-0.549	1010.20±521.46	899.00 (584.75-1442.50)	40-1911
	ObliqueRSF	0.599±0.051	0.601 (0.559-0.639)	0.491-0.697	0.240±0.041	0.233 (0.223-0.249)	0.169-0.388	1153.10±524.79	1222.50 (839.00-1561.75)	129-1929
	Penalized	**0.634±0.041**	0.635 (0.603-0.664)	0.555-0.730	0.225±0.029	0.221 (0.203-0.244)	0.159-0.308	916.27±554.06	813.00 (436.25-1271.75)	82-1951
	Ranger	0.629±0.036	0.635 (0.609-0.656)	0.527-0.695	**0.206±0.012**	0.205 (0.197-0.214)	0.185-0.243	703.37±511.13	625.00 (302.75-936.50)	35-1962
	Rfsrc	**0.634±0.032**	0.633 (0.615-0.652)	0.543-0.697	0.207±0.011	0.206 (0.200-0.213)	0.183-0.239	750.80±502.84	710.50 (253.25-1137.50)	33-1752
	Ridge	**0.650±0.042**	0.650 (0.626-0.674)	0.558-0.743	0.325±0.076	0.306 (0.269-0.378)	0.194-0.505	939.20±514.37	871.50 (492.75-1352.50)	150-1831
	Rpart	0.557±0.054	0.555 (0.526-0.588)	0.443-0.683	0.298±0.040	0.294 (0.270-0.337)	0.229-0.366	792.70±542.19	800.50 (297.00-1072.25)	44-1829
	Svm	0.609±0.066	0.606 (0.577-0.656)	0.416-0.748	0.437±0.062	0.434 (0.389-0.469)	0.291-0.560	1061.53±500.87	1034.50 (710.00-1451.75)	108-1968
	Xgboost (dart)	0.597±0.048	0.592 (0.570-0.633)	0.499-0.698	0.278±0.053	0.267 (0.244-0.299)	0.192-0.413	910.90±635.97	718.00 (260.50-1455.50)	31-1985
	Xgboost (gblinear)	0.617±0.056	0.626 (0.570-0.656)	0.500-0.699	0.253±0.021	0.247 (0.239-0.267)	0.214-0.304	1184.37±519.21	1133.50 (887.75-1709.25)	33-1911
	Xgboost (gbtree)	0.595±0.054	0.593 (0.551-0.642)	0.503-0.684	0.297±0.055	0.291 (0.249-0.353)	0.216-0.404	815.33±499.45	741.00 (435.00-1271.50)	47-1944
Models Boruta	**Blackboost**	0.583±0.053	0.579 (0.540-0.613)	0.497-0.706	0.418±0.111	0.453 (0.425-0.485)	0.197-0.527	**14.87±4.33**	14.00 (12.00-17.50)	8-24
	Cforest	0.610±0.055	0.616 (0.568-0.640)	0.485-0.738	0.211±0.022	0.209 (0.196-0.225)	0.174-0.265			
	Coxboost	0.601±0.054	0.614 (0.552-0.643)	0.496-0.696	0.252±0.038	0.246 (0.224-0.290)	0.188-0.321			
	Ctree	0.558±0.047	0.555 (0.521-0.585)	0.481-0.698	0.365±0.057	0.372 (0.318-0.416)	0.250-0.481			
	Elasticnet	0.593±0.048	0.594 (0.551-0.629)	0.496-0.673	0.329±0.095	0.307 (0.257-0.384)	0.204-0.552			
	Gbm	0.596±0.059	0.596 (0.555-0.620)	0.502-0.746	0.386±0.077	0.388 (0.330-0.452)	0.215-0.552			
	Glmboost	0.598±0.050	0.597 (0.568-0.636)	0.509-0.710	0.397±0.116	0.447 (0.231-0.480)	0.199-0.527			
	Lasso	0.597±0.053	0.606 (0.551-0.641)	0.483-0.687	0.353±0.100	0.339 (0.273-0.454)	0.176-0.559			
	ObliqueRSF	0.588±0.069	0.592 (0.532-0.664)	0.469-0.700	0.242±0.048	0.239 (0.211-0.267)	0.148-0.361			
	Penalized	0.605±0.046	0.605 (0.568-0.637)	0.508-0.716	0.231±0.032	0.225 (0.204-0.255)	0.171-0.297			
	Ranger	**0.619±0.052**	0.622 (0.589-0.654)	0.488-0.722	**0.209±0.017**	0.208 (0.198-0.217)	0.176-0.251			
	Rfsrc	0.608±0.051	0.599 (0.579-0.639)	0.485-0.739	0.220±0.021	0.216 (0.202-0.232)	0.184-0.278			
	Ridge	0.608±0.046	0.612 (0.574-0.641)	0.517-0.697	**0.207±0.008**	0.205 (0.200-0.215)	0.192-0.221			
	Rpart	0.563±0.059	0.553 (0.524-0.605)	0.465-0.739	0.302±0.044	0.305 (0.271-0.333)	0.231-0.391			
	Svm	0.553±0.055	0.557 (0.525-0.590)	0.407-0.640	0.450±0.062	0.452 (0.395-0.499)	0.355-0.566			
	Xgboost (dart)	0.588±0.054	0.578 (0.555-0.613)	0.501-0.709	0.285±0.054	0.273 (0.241-0.334)	0.198-0.406			
	Xgboost (gblinear)	0.586±0.055	0.604 (0.545-0.623)	0.487-0.677	0.254±0.030	0.250 (0.234-0.271)	0.211-0.335			
	Xgboost (gbtree)	0.589±0.061	0.572 (0.554-0.625)	0.486-0.751	0.295±0.066	0.285 (0.240-0.354)	0.195-0.428			
MLSeqSurv	**voomStackPrio1**	0.613±0.037	0.612 (0.587-0.637)	0.524-0.675	0.202±0.014	0.199 (0.194-0.209)	0.176-0.239	**16.27±4.81**	16.00 (13.00-19.25)	7-30
	voomStackPrio2	0.597±0.043	0.594 (0.556-0.635)	0.528-0.676	**0.197±0.010**	0.199 (0.191-0.204)	0.175-0.213			
	voomStackIPF1	0.613±0.035	0.613 (0.591-0.637)	0.524-0.694	**0.193±0.011**	0.192 (0.185-0.202)	0.166-0.214			
	voomStackIPF2	0.610±0.042	0.601 (0.577-0.651)	0.530-0.692	**0.193±0.012**	0.195 (0.187-0.201)	0.168-0.210			
	voomStackIPF3	0.609±0.042	0.603 (0.581-0.650)	0.527-0.687	**0.194±0.012**	0.196 (0.187-0.202)	0.168-0.212			
	voomStackIPF4	0.613±0.033	0.610 (0.591-0.636)	0.553-0.695	**0.192±0.010**	0.192 (0.186-0.201)	0.166-0.208			
	voomStackIPF5	0.608±0.044	0.601 (0.577-0.647)	0.527-0.692	**0.194±0.012**	0.196 (0.187-0.202)	0.168-0.213			
	voomStackIPF6	0.607±0.045	0.603 (0.581-0.642)	0.522-0.687	**0.194±0.012**	0.197 (0.187-0.203)	0.168-0.213			
	voomStackIPF7	**0.615±0.032**	0.613 (0.592-0.637)	0.561-0.696	**0.192±0.011**	0.192 (0.185-0.203)	0.165-0.210			
	voomStackIPF8	0.609±0.045	0.601 (0.577-0.653)	0.527-0.692	**0.193±0.013**	0.196 (0.187-0.203)	0.162-0.213			
	voomStackIPF9	0.608±0.045	0.603 (0.581-0.652)	0.522-0.687	**0.194±0.012**	0.197 (0.187-0.203)	0.168-0.213			

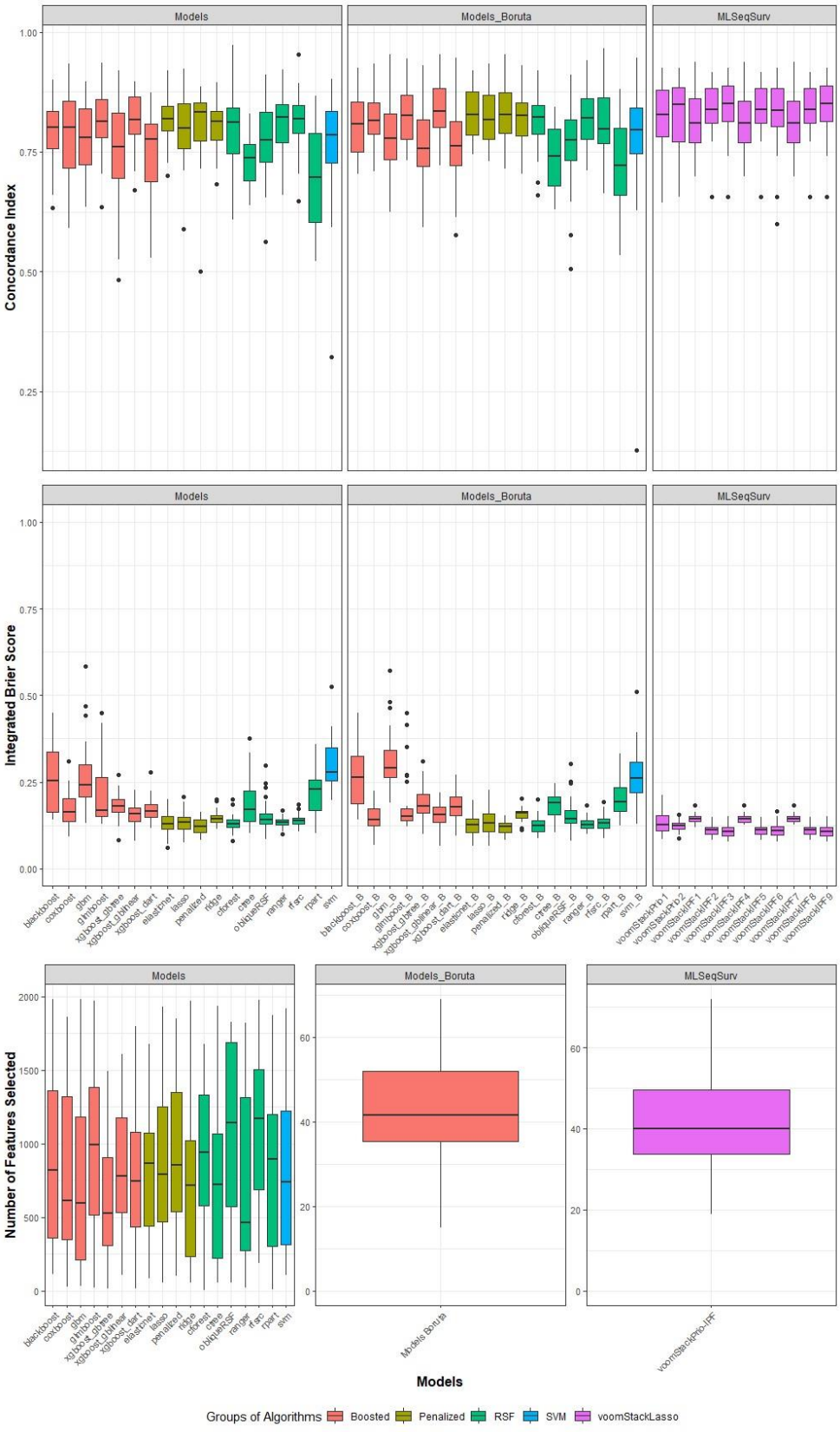


Figure 4.12. The concordance index, integrated Brier score, and the number of selected features for UVM.

Table 4.12. The summary statistics of concordance index, integrated Brier score and the number of features selected for UVM.

Groups of Algorithms	Models	Concordance Index			Integrated Brier Score			The Number of Features Selected		
		Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max	Mean±Sd. Deviation	Median (1st-3rd Quartile)	Min-Max
Models	**Blackboost**	0.789±0.068	0.802 (0.752-0.840)	0.633-0.900	0.258±0.096	0.253 (0.162-0.340)	0.140-0.450	882.90±583.12	817.50 (331.25-1406.00)	113-1981
	Cforest	0.798±0.071	0.811 (0.738-0.842)	0.608-0.972	0.131±0.024	0.129 (0.120-0.141)	0.079-0.200	916.10±468.87	940.50 (570.75-1368.75)	4-1678
	Coxboost	0.786±0.086	0.800 (0.715-0.858)	0.591-0.933	0.176±0.051	0.164 (0.135-0.208)	0.093-0.311	777.53±578.15	610.50 (320.75-1434.75)	26-1860
	Ctree	0.731±0.053	0.738 (0.689-0.768)	0.639-0.829	0.189±0.074	0.171 (0.135-0.231)	0.103-0.375	772.93±607.28	720.50 (218.25-1143.00)	56-1939
	Elasticnet	**0.819±0.056**	0.818 (0.789-0.851)	0.701-0.920	0.132±0.032	0.129 (0.114-0.152)	0.060-0.200	823.77±419.77	868.50 (414.00-1082.25)	81-1679
	Gbm	0.780±0.069	0.779 (0.722-0.846)	0.636-0.897	0.269±0.098	0.242 (0.203-0.309)	0.132-0.584	735.80±601.25	595.50 (201.75-1234.75)	32-1984
	Glmboost	**0.815±0.068**	0.813 (0.777-0.871)	0.636-0.935	0.218±0.094	0.168 (0.150-0.271)	0.130-0.450	1009.93±566.02	994.00 (478.25-1403.75)	18-1972
	Lasso	0.799±0.070	0.799 (0.755-0.854)	0.589-0.923	0.135±0.030	0.134 (0.113-0.148)	0.075-0.207	854.50±545.62	789.50 (458.75-1273.75)	57-1932
	ObliqueRSF	0.772±0.079	0.774 (0.721-0.836)	0.562-0.911	0.151±0.045	0.142 (0.127-0.162)	0.086-0.298	1116.40±560.43	1140.50 (553.50-1709.75)	54-1827
	Penalized	0.808±0.076	0.833 (0.765-0.853)	0.500-0.886	**0.122±0.022**	0.122 (0.105-0.142)	0.083-0.163	892.07±495.71	853.50 (494.50-1353.00)	102-1848
	Ranger	0.808±0.070	0.822 (0.757-0.852)	0.659-0.921	0.134±0.016	0.134 (0.127-0.143)	0.100-0.169	783.00±612.29	463.50 (266.75-1383.75)	21-1819
	Rfsrc	0.811±0.064	0.818 (0.781-0.850)	0.648-0.953	0.140±0.018	0.139 (0.129-0.147)	0.107-0.185	1102.63±534.90	1171.00 (661.00-1524.00)	186-1976
	Ridge	0.808±0.056	0.814 (0.769-0.842)	0.682-0.894	0.145±0.020	0.143 (0.133-0.154)	0.115-0.200	722.47±540.51	719.00 (229.00-1043.50)	57-1971
	Rpart	0.698±0.106	0.696 (0.596-0.794)	0.522-0.867	0.224±0.072	0.231 (0.162-0.258)	0.102-0.359	842.43±545.78	894.00 (244.50-1250.50)	11-1871
	Svm	0.759±0.117	0.785 (0.717-0.837)	0.322-0.902	0.303±0.071	0.279 (0.252-0.352)	0.198-0.524	834.00±572.32	737.50 (298.50-1230.00)	105-1920
	Xgboost (dart)	0.752±0.091	0.777 (0.682-0.813)	0.528-0.874	0.169±0.033	0.165 (0.149-0.186)	0.116-0.278	820.17±536.59	744.00 (391.25-1134.50)	14-1801
	Xgboost (gblinear)	**0.816±0.056**	0.817 (0.786-0.866)	0.670-0.897	0.154±0.032	0.158 (0.135-0.175)	0.081-0.226	830.43±447.23	780.50 (511.25-1218.75)	107-1611
	Xgboost (gbtree)	0.746±0.102	0.761 (0.689-0.834)	0.483-0.920	0.181±0.039	0.179 (0.162-0.201)	0.082-0.270	617.87±407.24	524.00 (278.25-932.75)	15-1496
Models Boruta	**Blackboost**	0.804±0.063	0.808 (0.747-0.865)	0.704-0.926	0.260±0.085	0.265 (0.182-0.336)	0.142-0.450	**43.57±12.03**	41.50 (34.75-52.25)	15-69
	Cforest	0.810±0.058	0.823 (0.784-0.848)	0.659-0.920	0.127±0.026	0.124 (0.104-0.142)	0.087-0.199			
	Coxboost	0.821±0.051	0.815 (0.786-0.853)	0.708-0.933	0.146±0.039	0.141 (0.125-0.174)	0.068-0.224			
	Ctree	0.736±0.066	0.740 (0.677-0.806)	0.629-0.843	0.183±0.036	0.190 (0.154-0.209)	0.105-0.247			
	Elasticnet	0.829±0.055	0.828 (0.783-0.881)	0.744-0.920	0.128±0.033	0.127 (0.103-0.147)	0.066-0.197			
	Gbm	0.776±0.069	0.778 (0.733-0.832)	0.625-0.953	0.312±0.086	0.290 (0.261-0.343)	0.191-0.572			
	Glmboost	0.825±0.060	0.825 (0.770-0.869)	0.733-0.944	0.185±0.085	0.150 (0.138-0.177)	0.122-0.450			
	Lasso	0.819±0.056	0.816 (0.773-0.868)	0.731-0.933	0.134±0.040	0.131 (0.106-0.158)	0.065-0.228			
	ObliqueRSF	0.769±0.088	0.775 (0.729-0.821)	0.505-0.911	0.155±0.047	0.143 (0.131-0.171)	0.079-0.302			
	Penalized	0.830±0.065	0.827 (0.787-0.882)	0.714-0.953	**0.119±0.019**	0.122 (0.103-0.135)	0.084-0.153			
	Ranger	0.818±0.063	0.820 (0.775-0.865)	0.711-0.941	0.129±0.018	0.127 (0.116-0.141)	0.100-0.183			
	Rfsrc	0.811±0.075	0.797 (0.766-0.866)	0.663-0.965	0.134±0.025	0.130 (0.117-0.145)	0.086-0.193			
	Ridge	0.818±0.059	0.826 (0.775-0.857)	0.704-0.929	0.157±0.018	0.161 (0.146-0.166)	0.112-0.203			
	Rpart	0.721±0.090	0.722 (0.657-0.803)	0.534-0.880	0.203±0.053	0.192 (0.165-0.239)	0.124-0.332			
	Svm	0.770±0.146	0.796 (0.728-0.845)	0.128-0.947	0.269±0.080	0.261 (0.218-0.314)	0.129-0.511			
	Xgboost (dart)	0.767±0.092	0.763 (0.718-0.819)	0.577-0.947	0.181±0.039	0.178 (0.153-0.207)	0.095-0.270			
	Xgboost (gblinear)	**0.839±0.061**	0.834 (0.797-0.883)	0.721-0.953	0.155±0.035	0.157 (0.130-0.179)	0.066-0.219			
	Xgboost (gbtree)	0.766±0.074	0.757 (0.717-0.821)	0.592-0.929	0.190±0.049	0.180 (0.161-0.216)	0.100-0.309			
MLSeqSurv	**voomStackPrio1**	0.827±0.063	0.828 (0.777-0.880)	0.644-0.926	0.133±0.033	0.127 (0.108-0.158)	0.086-0.212	**42.20±12.95**	40.00 (33.00-50.50)	19-72
	voomStackPrio2	0.828±0.070	0.849 (0.766-0.887)	0.655-0.926	0.123±0.015	0.123 (0.114-0.131)	0.088-0.157			
	voomStackIPF1	0.817±0.063	0.810 (0.764-0.864)	0.698-0.937	0.144±0.013	0.143 (0.135-0.152)	0.120-0.183			
	voomStackIPF2	**0.840±0.055**	0.839 (0.807-0.883)	0.655-0.916	**0.111±0.016**	0.112 (0.099-0.121)	0.083-0.150			
	voomStackIPF3	**0.841±0.059**	0.851 (0.810-0.888)	0.655-0.926	**0.108±0.018**	0.107 (0.095-0.119)	0.077-0.151			
	voomStackIPF4	0.817±0.062	0.810 (0.763-0.859)	0.698-0.937	0.144±0.013	0.143 (0.134-0.153)	0.126-0.183			
	voomStackIPF5	**0.840±0.055**	0.839 (0.807-0.883)	0.655-0.916	**0.111±0.016**	0.112 (0.099-0.121)	0.083-0.150			
	voomStackIPF6	0.830±0.072	0.836 (0.798-0.885)	0.600-0.926	**0.111±0.020**	0.110 (0.097-0.123)	0.077-0.165			
	voomStackIPF7	0.817±0.062	0.810 (0.764-0.859)	0.698-0.937	0.144±0.012	0.143 (0.135-0.152)	0.126-0.183			
	voomStackIPF8	**0.840±0.055**	0.839 (0.807-0.883)	0.655-0.916	**0.111±0.016**	0.112 (0.099-0.121)	0.083-0.150			
	voomStackIPF9	**0.841±0.059**	0.851 (0.810-0.888)	0.655-0.926	**0.108±0.018**	0.107 (0.095-0.119)	0.077-0.151			

features was xgboost (with booster= “gbtree”) (617.87±407.24), while the algorithm with the highest mean number of features was obliqueRSF (1116.40±560.43).

4.2. Super Lists

Within the study's scope is a plan to compare the model performances of 47 algorithms, including the newly developed ones. Due to the extensive number of methods, diverse datasets, and multiple performance measures, determining the best methods in every aspect is challenging. Therefore, the rank aggregation method was employed. Rank aggregation is an optimization method capable of combining ordered lists. Utilizing this approach, separate assessments were conducted based on the number of selected features, concordance index, and integrated Brier score. As a result of these evaluations, super lists were generated for each performance measure, ranking methods from the best-performing to the worst-performing (Figure 4.13, Figure 4.14, and Figure 4.15). Super lists have been generated based on the colors and numbers assigned to the compared methods:

voomStackPrio1	1
voomStackPrio2	2
voomStackIPF1	3
voomStackIPF2	4
voomStackIPF3	5
voomStackIPF4	6
voomStackIPF5	7
voomStackIPF6	8
voomStackIPF7	9
voomStackIPF8	10
voomStackIPF9	11

blackboost	1
coxboost	2
gbm	3
glmboost	4
xgboost_gbtree	5
xgboost_gblinear	6
xgboost_dart	7
elasticnet	1
lasso	2
penalized	3
ridge	4
cforest	1
ctree	2
obliqueRSF	3
ranger	4
rfsrc	5
rpart	6
svm	1

blackboost_B	1
coxboost_B	2
gbm_B	3
glmboost_B	4
xgboost_gbtree_B	5
xgboost_gblinear_B	6
xgboost_dart_B	7
elasticnet_B	1
lasso_B	2
penalized_B	3
ridge_B	4
cforest_B	1
ctree_B	2
obliqueRSF_B	3
ranger_B	4
rfsrc_B	5
rpart_B	6
svm_B	1

In Figure 4.13, a super list has been compiled based on the concordance index. Initially, the concordance index results for each dataset were arranged in descending order on separate lines. Running the rank aggregation algorithm across these 12 datasets formed the super list displayed in the bottom row. Among the top 20 methods

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
ACC	1	1	4	5	1	4	3	1	9	4	6	6	3	4	1	6	5	3	1	1	3	3	2	4	2	7	4	5	2	3	7	5	2	3	4	7	10	5	11	8	1	6	6	2	2	2	1
CESC	4	3	1	3	4	9	1	6	4	1	2	1	4	5	7	4	2	8	10	5	11	1	4	6	2	2	3	2	4	3	3	1	5	1	5	7	5	7	3	1	3	6	1	6	2	2	6
ESCA	4	9	1	3	6	7	4	5	10	8	11	1	5	1	3	2	5	4	2	3	7	6	3	6	4	7	5	1	2	6	5	2	3	1	2	2	2	1	4	1	1	4	4	3	1	3	6
GBM	4	1	3	3	1	2	5	2	3	4	2	1	6	1	3	4	7	8	1	10	11	5	2	4	4	2	5	4	7	1	4	6	6	1	2	1	1	5	6	3	6	3	9	5	7	3	2
KIRC	1	4	5	4	1	3	1	6	4	1	4	3	2	2	1	3	2	3	6	9	4	5	4	1	7	4	10	8	1	11	5	3	7	2	5	2	5	3	3	6	1	7	6	2	6	2	1
KIRP	1	3	2	5	4	2	4	3	1	4	1	9	2	6	4	3	3	1	7	2	1	5	3	4	4	5	1	5	7	10	1	3	4	7	2	3	6	5	11	2	8	1	6	6	6	2	1
LAML	2	5	2	1	4	1	4	3	1	3	6	9	6	3	4	4	5	3	2	1	3	4	2	1	10	3	4	2	1	1	7	5	4	7	11	7	1	6	5	5	3	8	6	6	2	2	1
LGG	2	4	4	1	2	1	3	4	5	4	2	4	3	9	1	3	2	6	1	1	4	1	5	4	7	10	6	6	5	8	5	11	1	7	3	3	1	7	5	3	2	6	3	6	2	2	1
MESO	3	9	6	4	1	3	6	4	6	1	4	3	1	1	5	4	1	1	2	4	4	5	2	2	1	1	3	2	3	10	11	8	7	7	4	5	2	5	3	7	6	2	5	3	2	6	1
PAAD	6	9	4	6	6	3	1	3	3	1	2	4	4	4	2	4	2	1	7	3	8	11	4	10	5	1	1	1	5	2	4	1	2	3	5	1	7	3	5	2	3	5	7	6	6	2	1
SARC	4	1	5	3	4	1	4	6	4	2	9	3	1	3	6	1	1	4	1	5	2	10	5	4	7	11	8	3	2	3	4	2	7	2	3	5	1	5	7	3	6	1	2	6	2	6	1
UVM	5	11	4	7	10	6	8	3	1	2	1	4	2	2	1	4	4	3	6	9	6	4	5	5	1	3	4	4	1	2	1	1	2	3	3	3	1	3	7	5	1	7	5	2	2	6	6
	4	**4**	**1**	**1**	**3**	**5**	**9**	**6**	**4**	**2**	**4**	**2**	**3**	**3**	**1**	**4**	**3**	**1**	**2**	**6**	**5**	**1**	**4**	**10**	**7**	**4**	**8**	**5**	**2**	**1**	**11**	**5**	**7**	**2**	**1**	**5**	**3**	**7**	**3**	**1**	**6**	**3**	**6**	**6**	**2**	**2**	**1**
Super List	ranger	ridge	elasticnet	cforest	voomStackIPF1	rfsrc	voomStackIPF7	voomStackIPF4	ranger_B	coxboost	ridge_B	lasso	penalized_B	gbm	elasticnet_B	glmboost	penalized	blackboost	lasso_B	xgboost_gblinear	rfsrc_B	cforest_B	voomStackIPF2	voomStackIPF8	voomStackIPF5	glmboost_B	voomStackIPF6	voomStackIPF3	coxboost_B	voomStackPrio1	voomStackIPF9	xgboost_gbtree	xgboost_dart	voomStackPrio2	blackboost_B	xgboost_gbtree_B	obliqueRSF	xgboost_dart_B	obliqueRSF_B	svm	xgboost_gblinear_B	gbm_B	rpart_B	rpart	ctree_B	ctree	svm_B

Figure 4.13. The ranking of survival algorithms based the concordance index.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
ACC	3	9	3	6	4	10	7	4	3	1	5	11	8	5	3	1	4	2	5	6	2	1	5	7	3	7	6	5	2	4	2	6	6	2	4	1	1	4	1	4	1	2	3	3	1	1	2
CESC	6	3	9	7	2	8	4	5	10	11	3	5	1	4	4	3	4	1	5	1	2	6	4	3	6	5	7	3	2	7	5	1	2	6	1	6	2	2	2	4	4	1	1	1	3	3	1
ESCA	1	3	4	3	4	2	5	4	2	6	3	9	7	8	4	5	2	10	11	1	4	5	6	6	3	3	2	1	6	6	7	7	5	5	1	3	2	2	1	3	1	4	1	2	4	1	1
GBM	1	3	3	4	5	1	4	2	4	3	2	3	5	7	8	3	9	4	6	5	10	11	6	2	6	1	6	6	5	7	5	7	4	2	1	3	1	2	3	1	1	2	4	2	4	1	1
KIRC	1	3	3	4	5	4	2	5	1	7	4	10	8	11	5	2	3	6	9	2	3	1	3	4	6	5	6	7	7	5	4	1	1	4	3	2	6	2	2	6	1	3	1	4	2	1	1
KIRP	10	3	9	4	6	7	2	5	8	11	3	1	4	4	3	5	5	1	3	5	3	6	2	6	7	7	5	2	1	2	2	1	4	1	1	6	4	4	6	4	3	1	3	2	1	2	1
LAML	1	3	1	5	4	4	3	5	10	7	4	11	8	5	2	9	3	6	4	3	1	2	2	6	3	6	5	7	3	5	6	6	7	4	3	2	2	1	1	1	2	2	1	4	4	1	1
LGG	2	2	3	3	4	4	1	3	5	3	1	5	6	6	2	4	7	10	8	5	11	9	6	3	6	6	5	7	5	7	4	1	3	3	4	2	1	2	1	1	1	2	2	1	1	4	4
MESO	3	3	4	1	1	2	5	2	4	5	3	6	3	10	8	11	6	4	7	5	1	4	7	2	6	9	3	5	7	5	6	4	6	2	1	3	2	2	1	2	3	4	4	1	1	1	1
PAAD	3	3	1	4	2	5	7	2	5	8	4	11	10	4	9	3	6	1	1	4	5	6	2	4	3	6	3	7	6	5	6	5	7	2	3	1	3	2	1	2	4	2	1	4	1	1	1
SARC	6	9	3	4	5	10	7	8	11	2	1	1	4	4	5	4	1	5	3	3	3	3	2	6	6	2	7	7	5	5	6	6	1	2	4	1	2	2	2	4	3	3	1	4	1	1	3
UVM	5	11	7	4	10	8	3	3	2	1	1	4	1	1	1	2	5	4	2	5	9	3	6	4	2	3	6	6	3	4	7	2	7	5	2	4	2	5	6	4	6	1	1	1	3	1	3
	3	**1**	**3**	**4**	**7**	**8**	**4**	**4**	**5**	**10**	**5**	**11**	**5**	**2**	**9**	**3**	**6**	**1**	**2**	**3**	**6**	**2**	**6**	**7**	**5**	**7**	**4**	**6**	**4**	**6**	**1**	**2**	**2**	**1**	**1**	**2**	**2**	**4**	**3**	**3**	**3**	**1**	**1**	**4**	**1**	**1**	**5**
Super List	penalized_B	cforest	penalized	voomStackIPF2	voomStackIPF5	voomStackIPF6	ranger	ranger_B	rfsrc	voomStackIPF8	voomStackIPF3	voomStackIPF9	rfsrc_B	voomStackPrio2	voomStackIPF7	voomStackIPF1	voomStackIPF4	voomStackPrio1	coxboost_B	obliqueRSF	xgboost_gblinear_B	coxboost	xgboost_gblinear	xgboost_dart_B	xgbooost_gbtree	xgbooost_dart	ridge	rpart_B	ridge_B	rpart	elasticnet	ctree_B	lasso_B	elasticnet_B	cforest_B	lasso	ctree	glmboost	gbm	gbm_B	obliqueRSF_B	blackboost	blackboost_B	glmboost_B	svm	svm_B	xgboost_gbtree_B

Figure 4.14. The ranking of survival algorithms based the integrated Brier score.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ACC	MLSeqSurv	Models_Boruta	6	3	2	4	4	7	5	4	2	1	6	2	1	5	1	3	1	3
CESC	MLSeqSurv	Models Boruta	1	5	4	6	4	3	1	4	1	3	7	5	2	2	3	1	2	6
ESCA	Models Boruta	MLSeqSurv	3	4	1	4	2	4	3	1	3	2	5	7	1	6	1	2	6	5
GBM	Models Boruta	MLSeqSurv	4	4	5	3	4	3	6	1	2	5	1	2	2	1	1	7	3	6
KIRC	MLSeqSurv	Models Boruta	6	2	4	4	1	3	3	2	5	2	4	5	1	3	7	1	1	6
KIRP	Models Boruta	MLSeqSurv	6	1	2	3	2	5	7	4	3	2	3	1	1	4	5	4	1	6
LAML	Models Boruta	MLSeqSurv	7	5	4	2	2	1	4	6	4	5	1	3	1	6	3	2	1	3
LGG	Models Boruta	MLSeqSurv	7	4	2	2	1	5	6	3	1	1	4	5	3	2	3	1	4	6
MESO	Models Boruta	MLSeqSurv	4	5	2	7	6	5	1	3	4	3	4	1	2	1	1	6	2	3
PAAD	MLSeqSurv	Models Boruta	5	4	3	2	6	4	3	4	1	5	2	2	7	1	1	1	6	3
SARC	Models Boruta	MLSeqSurv	4	5	2	6	5	2	3	7	3	4	2	4	1	1	1	1	3	6
UVM	MLSeqSurv	Models Boruta	5	4	3	2	2	4	7	1	6	1	6	2	1	3	1	4	5	3
Super List	**Models Boruta**	**MLSeqSurv**	**4**	**2**	**6**	**4**	**4**	**3**	**7**	**5**	**2**	**2**	**5**	**1**	**1**	**1**	**3**	**1**	**3**	**6**
	Models Boruta	MLSeqSurv	ranger	ctree	rpart	ridge	glmboost	penalized	xgboost_dart	xgboost_gbtree	coxboost	lasso	rfsrc	cforest	elasticnet	blackboost	gbm	svm	obliqueRSF	xgboost_gblinear

Figure 4.15. The ranking of survival algorithms based the number of selected features.

according to the concordance index in this list are the following: ranger, ridge, elasticnet, cforest, voomStackIPF1, rfsrc, voomStackIPF7, voomStackIPF4, ranger_B, coxboost, ridge_B, lasso, penalized_B, gbm, elasticnet_B, glmboost, penalized, blackboost, lasso_B, xgboost (gblinear).

In Figure 4.14, a super list has been compiled based on the integrated Brier score. Initially, the integrated Brier score results for each dataset were arranged on separate lines from smallest to largest. Running the rank aggregation algorithm across these 12 datasets formed the super list displayed in the bottom row. Among the top 20 methods according to the integrated Brier score in this list are the following, listed in order: penalized_b, cforest, gbm, voomStackIPF2, voomStackIPF5, voomStackIPF6, ranger, ranger_B, rfsrc, voomStackIPF8, voomStackIPF3, voomStackIPF9, rfsrc_B, voomStackPrio2, voomStackIPF7, voomStackIPF1, voomStackIPF4, voomStackPrio1, coxboost_b, obliqueRSF.

In Figure 4.15, a super list has been generated based on the number of selected features. Initially, lists were created for each dataset, ranging from using the least

features to using the most features. In the resulting super list, it was chosen the least number of features.

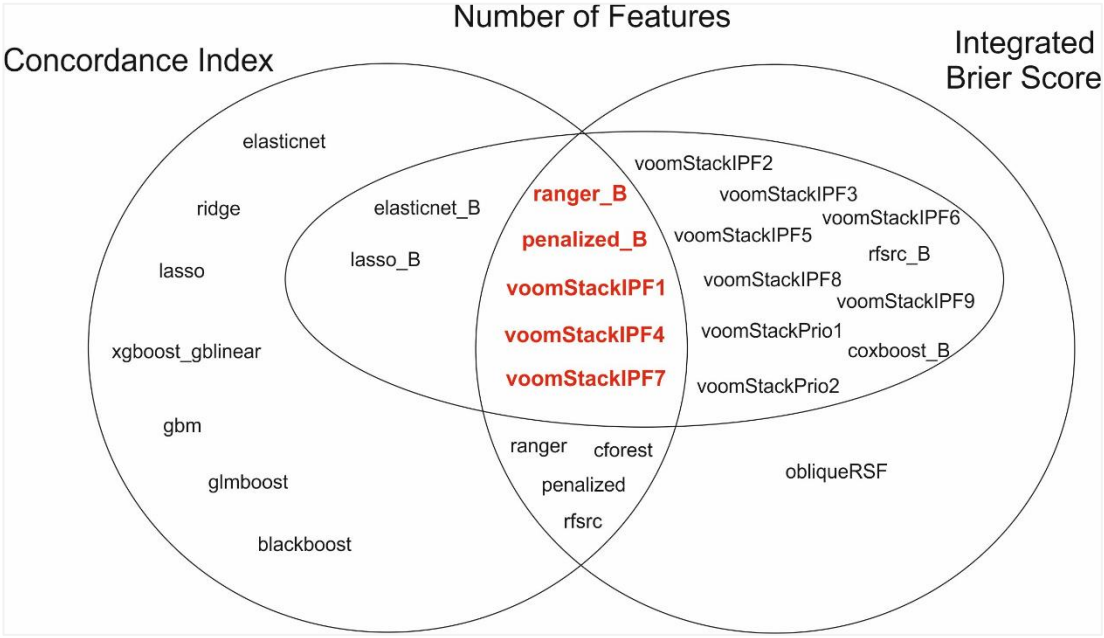


Figure 4.16. A Venn diagram illustrating optimal practices concerning concordance index, integrated Brier score, and the number of selected features.

Upon scrutinizing the survival models in this study solely based on a high concordance index, it was evident that the lasso, elastic net, ridge, xgboost (gblinear), gbm, glmboost, and blackboost algorithms, employing internal feature selection, exhibited commendable performance. The obliqueRSF, which also utilizes internal feature selection, was the sole model displaying a low integrated Brier score. Models with both a high concordance index and a low integrated Brier score include cforest, ranger, rfsrc, and penalized—each utilizing internal feature selection. Nevertheless, despite their high performance in concordance index and integrated Brier score, these models tend to incorporate excessive features. Elastic net and lasso models utilizing Boruta feature selection exhibited low features and a high concordance index value. However, their integrated Brier scores were also high. On the other hand, voomStackIPF2, voomStackIPF3, voomStackIPF5, voomStackIPF6, voomStackIPF8, voomStackIPF9, voomStackPrio1, voomStackPrio2, rfsrc, and coxboost algorithms, when employing Boruta feature selection, displayed a low feature count and a low integrated Brier score value. Despite this, their concordance index values were not notably high. Methods striking a balance between a high concordance index, low

integrated Brier score, and a modest feature count include voomStackIPF1, voomStackIPF4, voomStackIPF7, ranger, and penalized algorithms—all utilizing Boruta feature selection (Figure 4.16).

4.3. Computational Time

The execution times of the algorithms for MESO, SARC, and LGG data are presented in minutes in Table 4.13. Upon examination, it is evident that voomStackLasso algorithms typically complete calculations in significantly less time than existing algorithms. This characteristic enhances the practical utility of the newly developed algorithms.

Table 4.13. The summary statistics of the computational time for MESO, SARC, and LGG.

Groups of Algorithms	Models	MESO	SARC	LGG
Models	**Blackboost**	60.69	55.94	102.08
	Cforest	195.91	201.36	456.48
	Coxboost	49.73	90.32	192.55
	Ctree	18.23	24.9	39.9
	Elasticnet	1.54	2.79	3.51
	Gbm	5.01	23.4	48.37
	Glmboost	18.56	55.71	109.59
	Lasso	1.76	2.75	3.58
	ObliqueRSF	58.84	509.18	1064.64
	Penalized	2.69	6.39	8.33
	Ranger	2.17	72.07	419.81
	Rfsrc	2.19	11.05	22.17
	Ridge	1.35	2.89	5.00
	Rpart	1.54	3.44	4.01
	Svm	2.01	52.58	155.51
	Xgboost (dart)	1.06	4.54	5.7
	Xgboost (gblinear)	0.97	3.07	3.7
	Xgboost (gbtree)	1.1	4.31	7.1
Models Boruta	**Blackboost**	12.12	16.40	202.12
	Cforest	25.37	85.65	1052.77
	Coxboost	5.5	7.65	52.99
	Ctree	0.93	0.92	6.35
	Elasticnet	0.78	0.54	2.06
	Gbm	0.71	0.63	5.31
	Glmboost	6.97	11.48	104.63
	Lasso	0.72	0.55	2.06
	ObliqueRSF	26.21	66.44	243.68
	Penalized	0.39	0.52	1.47
	Ranger	5.85	99.52	544.57
	Rfsrc	2.2	17.71	31.12
	Ridge	0.75	0.54	2.03
	Rpart	1.18	2.01	2.08
	Svm	6.94	51.86	193.7
	Xgboost (dart)	2.39	4.13	3.71
	Xgboost (gblinear)	2.21	3.77	3.24
	Xgboost (gbtree)	2.41	4.02	3.7
MLSeqSurv	**voomStackPrio1**	0.59	0.94	13.48
	voomStackPrio2	0.58	0.85	13.36
	voomStackIPF1	0.41	0.56	5.73
	voomStackIPF2	0.72	0.45	9.82
	voomStackIPF3	1.36	0.48	48.88
	voomStackIPF4	0.42	0.53	7.24
	voomStackIPF5	0.77	0.51	11.86
	voomStackIPF6	0.99	0.44	50.64
	voomStackIPF7	0.43	0.53	5.40
	voomStackIPF8	0.66	0.41	4.56
	voomStackIPF9	1.13	0.34	37.78

The values in the table are calculated in minutes. Mean values are given.

5. DISCUSSION

Previously conducted with low-dimensional clinical data in earlier studies, survival analyses have undergone a transformative shift toward integrating large-dimensional gene expression data, such as RNA-seq. This shift reflects the advancing landscape of precision medicine, emphasizing personalized diagnoses and treatments over-generalized approaches applied uniformly to all patients. Researchers are using genetic data to diagnose and treat diseases because they have realized that different people respond differently to the same treatments and that diseases progress differently in different ways in the same individuals. This paradigm shift towards leveraging genetic information underscores the significance of tailoring medical interventions based on an individual's unique genetic makeup, ushering in a new era of targeted and more effective healthcare strategies.

Machine learning methods have been devised for survival analyses on high-dimensional RNA-seq data. In the context of this thesis, we compared these established methods in the literature with novel approaches we developed ourselves. The criteria employed for model comparisons encompassed the concordance index and integrated Brier score. Performance evaluation in survival analyses typically involves two main aspects: discrimination and calibration.

In survival analyses, the concordance index is the most widely used discrimination metric. Listing anticipated risk scores and actual results is the first step, and then the alignment of these rankings is compared. Concordance occurs when an individual who experiences an event at the start of the study period is given a greater predicted risk score than an individual who experiences the event at the end of the study period or never during the study period. This index represents the probability that two randomly chosen subjects have correctly ranked risk estimates. Despite its common use, the concordance index has drawbacks. It relies solely on the ranks of predicted values, potentially inflating the index for models with inaccurate predictions compared to competing models with more accurate predictions (245). Furthermore, when introducing new statistically and clinically significant variables to the model, the concordance index becomes less reliable, as it is insensitive to such additions (246). Additionally, its interpretation is limited due to the amalgamation of sensitivity and selectivity concepts (247).

Alternatively, the Brier score shows up as a complete performance measure that includes both calibration and discrimination. With respect to the predicted survival probability, this score measures the mean squared error. It can calculate a general error measurement over all time points and be applied to a particular time point (*t*). But in order to compute the Brier score, baseline estimation is required, and different approaches can produce different scores. In this study, we employed the Brier score developed by Graf et al. (219), eliminating the need for baseline estimation. This method categorizes test data into two groups based on the training model, estimating the risk-free probability for each sample from the Kaplan-Meier estimate relevant to the group.

Models were additionally assessed based on the number of features they selected. For models beyond the approaches we devised, analyses conducted using the **mlr3proba** package (215) revealed that the internal feature selection algorithms needed to be sufficiently sparse in their selection, encompassing a broad array of features. Consequently, the Boruta feature selection algorithm was applied in subsequent analyses, addressing the need for more focused feature selection. In this case, although models were created with fewer features, it has been observed that there is little difference in model performances. Moreover, previous studies experimenting with various feature selection methods for survival data have also indicated no difference in model performance (52,248).

Upon reviewing the concordance index results, it is noteworthy that while the internal feature selection and Boruta feature selection outcomes of algorithms in the literature generally yield similar results, distinct algorithms exhibit higher performance on different datasets. For instance, in evaluations based on concordance index results for methods in the literature employing internal feature selection, the ranger algorithm demonstrated superior performance in ESCA and GBM data, the ridge algorithm excelled in PAAD and SARC data, and the elastic net algorithm outperformed in MESO data. Notably, in the LAML data, the coxboost algorithm performed the best, while in the ACC data, the cforest algorithm performed exceptionally well. After scrutinizing the literature's approaches, it became evident that ctree and rpart methods generally yielded lower accuracy across all datasets. Comparative studies utilizing these algorithms concluded that different methods performed better in diverse studies despite generating similar performance results. In

a study encompassing ten different cancer datasets, Cox Proportional Hazards with Ridge penalty, Random Survival Forests, Gradient Boosting for Survival Analysis with a CoxPh loss function, linear and kernel Support Vector Machines generally delivered comparable concordance index results, particularly in higher-dimensional data. Cox proportional hazards demonstrated suboptimal performance in some high-dimensional sets (249). Numerous studies have consistently showcased the high-performance outcomes of the random survival forest algorithm (250–252). While xgboost and random survival forest algorithms demonstrated similar performance in certain studies (253), the xgboost algorithm outperformed in others (254,255). Furthermore, studies are highlighting the high performance of the elasticnet algorithm (256), the gbm algorithm (257), and the blackboost algorithm (53). In comparisons of survival algorithms, the svm algorithm either showed results comparable to other methods or exhibited lower performance outcomes (258).

When scrutinizing the results of the integrated Brier score, it is observed that penalized, ranger, rfsrc, and cforest algorithms, among the survival algorithms in the literature, consistently yield high-performance outcomes. Conversely, blackboost, elasticnet, gbm, lasso, and svm algorithms tend to exhibit notably lower performances. It's important to note that while algorithms with low integrated Brier score often demonstrate high concordance index performances, relying solely on the concordance index for model performance evaluation may not suffice for a comprehensive assessment. This issue was also highlighted by Hermann et al., (53) in their article, which argues that the selection of a performance measure can have a significant effect on the assessment of the performance of a method. While many studies traditionally measure survival model performances with the concordance index, Hermann et al. (53) argued that the cindex is not an ideal measure as it solely assesses discrimination. They proposed that the integrated Brier score is a more accurate measure. Additionally, Hermann et al. (50) highlighted that if the study's goal is risk classification, using the concordance index for interpretability is more accurate, whereas for prognostic accuracy, the integrated Brier score is the preferred metric.

When assessed in terms of the number of variables incorporated into the model, it becomes evident that the means and standard deviations of the selected variables in algorithms employing internal feature selection methods are notably high, indicating a lack of sparsity in these methods. However, upon applying Boruta feature selection

to these algorithms, models could be constructed with significantly fewer variables. In addition, Spooner et al. (52)compared model performance with survival algorithms and with feature selection methods and found that feature selection did not result in significant changes in model performance. This study observed that when both Boruta feature selection and internal feature selection were applied to survival algorithms in the literature, the model's performance was not significantly altered based on the concordance index and integrated Brier score. It can be stated that, in some datasets, model performances with Boruta feature selection are relatively lower than those with integrated feature selection. However, it is noteworthy that, in models where Boruta feature selection is applied, the mean and standard deviation of the selected feature numbers are considerably lower.

When the models are evaluated in terms of calculation times, it is noteworthy that boosting methods such as blackboost take a very long time to model. In addition, it has been observed that penalized Cox regression models such as lasso, ridge, elasticnet can model very quickly.

A comprehensive study was carried out, utilizing 12 real RNA-seq survival datasets, to assess the proposed methodologies' efficacy and compare their performance with that of other survival algorithms. Remarkably good results were achieved when applied to real data. Notably, three of the recently formulated voomStackIPF approaches (voomStackIPF1, voomStackIPF4, and voomStackIPF7) demonstrated comparable or slightly superior results compared to the most favorable results attained by established methods documented in the literature for the analysis of RNA-seq survival data. These models demonstrated a high concordance index and a low Brier score, showcasing the capability to generate models swiftly with minimal variables. In addition, the models were also evaluated for how long it took to achieve results. Upon further analysis, it became clear that the proposed algorithms delivered results in a fraction of the time as the existing algorithms described in the literature.

The new algorithms used in this study are the first to integrate logCPM and voom transformation-derived sample weights into survival algorithms for the first time. While voom transformation has previously been used in differential expression (69), classification (71), and clustering (72) analysis of RNA-seq data, this is the first study to use voom transform in survival algorithms. This study shows impressive performance and provides sparse results.The accuracy of the voom transformation to

model the mean-variance relationship in RNA seq data is thought to be critical to the success of our novel survival algorithms. Additionally, the weights obtained through the voom transformation confer advantages, such as accommodating samples with varying sorting depths and mitigating the impact of low-quality samples.

The stacking algorithm can alter the structure of survival data, converting it into a classification data format. Consequently, all algorithms designed for classification issues can be extended to address survival problems. Hence, the stacking algorithm, renowned for its high-performance outcomes when applied to low-dimensional data (61), was introduced for the first time to RNA-seq data in this study. In the context of this thesis, the stacking algorithm facilitated the utilization of priority-Lasso and IPF-Lasso classification algorithms, incorporating sample weights, for the inaugural analysis of RNA-seq survival data. Integration with various classification algorithms is achievable through the stacking algorithm. Rather than being confined to a restricted set of survival algorithms, researchers can now employ numerous classification algorithms, numbering in the hundreds drawn from existing literature and incorporating newly developed methods for enhanced analysis. Thus, the stacking algorithm and other established classification algorithms in the literature can now be applied in the survival data analysis.

In this thesis study, several key factors contributed to the success of the new algorithms used to analyze the survival data for RNA-seq: (i) the application of the potent voom transformation algorithm to the data, (ii) the transformation of the intricate structure of survival data into a simplified classification data structure through stacking algorithms, (iii) the utilization of priority-Lasso and IPF Lasso algorithms capable of analyzing the block structure of variables with diverse data structures obtained through stacking, leading to more precise results, (iv) the modeling approach involves applying distinct weights to individual samples in the RNA-seq data, and (v) the Boruta algorithm, known for its effectiveness in selecting important variables, has been integrated into newly developed algorithms. The combination of these robust approaches has led to the development of two different survival algorithms: high performance, sparse, and efficient modeling algorithms, which have made significant contributions to the literature.

The developed algorithms extend beyond conventional survival analysis; they also demonstrate exceptional proficiency in biomarker discovery. Their ability to

perform both survival assays and biomarker assays using gene expression data at the same time is of great importance to healthcare professionals, helping them to better diagnose patients. Creating a decision support system for clinical diagnosis, driven by identifying the most pertinent genes associated with a disease, empowers clinicians to make more accurate diagnoses promptly. This, in turn, facilitates the development of personalized treatments, enhances patients' quality of life, and positively contributes to the country's economy.

MLSeqSurv, which was created for this thesis, allows researchers to perform survival assays on the RNA-seq dataset using existing survival algorithms in the literature as well as new algorithms. This package makes it easy to create individual survival graphs so you can perform your own analysis without having to spend a lot of time coding.

The newly developed algorithms, voomStackPrio, and voomStackIPF, introduced within the context of this study, were implemented on RNA-seq data. However, these algorithms can be extended for future investigations to analyze other high-dimensional datasets such as microarray, proteomics, and metabolomics by customizing the pre-processing steps. It is anticipated that in other high-dimensional data settings, similar to RNA-seq, these algorithms would yield high-performance results, particularly in the context of survival analyses. Additionally, in this study, only protein-coding genes have been considered. However, in future studies, non-coding genes may also be included in the analysis. Similarly, this study utilized bulk RNA-seq data; however, the pre-processing step can be adapted and implemented for survival analyses of single-cell RNA-seq data.

In this study, survival prediction was achieved by looking at data with heterogeneous structures due to stacking through block structured lasso algorithms. Future survival algorithms could be created using multiple kernel algorithms where different types of data are evaluated in different cores. Moreover, studies have demonstrated performance improvements when combining RNA-seq data with clinical data or other omics data like microarray, metabolomics, and proteomics. This situation aligns well with the voomStackIPF and voomStackPrio algorithms developed in this study, which were designed to evaluate different data types in distinct blocks. For instance, in scenarios involving clinical + RNA-seq data, post voom transformation, and stacking, clinical data could be analyzed in one block while RNA-

seq data is assessed in another. Similarly, for RNA-seq + metabolomics + proteomic data, each data type could be processed and stacked separately according to its structure, allowing for analysis in individual blocks—one for RNA-seq data, another for metabolomics data, and a third for proteomic data.

6. CONCLUSION

In this paper, we introduced two new algorithms to the literature: voom transformation and stacking algorithm, as well as block-based lasso algorithms for RNA-seq survival analysis. These algorithms solve problems such as high dimensionality, high collinearity, and heterogeneity in RNA-seq data, tackling survival problems by conceptualizing them as classification problems. The algorithms developed have comparable or better model performance compared to other techniques described in the literature, showing their efficiency in building models with minimal features. In addition, these algorithms are much faster than existing algorithms in terms of computational time, delivering results in a fraction of the time.

On the basis of these results, voomStackLasso algorithms serve as a viable alternative to other survival algorithms used in the analysis of RNA-seq datasets.

7. REFERENCES

1. Fielden MR, Zacharewski TR. Challenges and limitations of gene expression profiling in mechanistic and predictive toxicology. Toxicol Sci. 2001;60(1):6–10.

2. Richard C. Analysis of cell division parameters and cell cycle gene expression during the cultivation of Arabidopsis thaliana cell suspensions. J Exp Bot. 2001;52(361):1625–33.

3. Bertucci F, Finetti P, Rougemont J, Charafe-Jauffret E, Nasser V, Loriod B, et al. Gene expression profiling for molecular characterization of inflammatory breast cancer and prediction of response to chemotherapy. Cancer Res. 2004;64(23):8558–65.

4. Lowe R, Shirley N, Bleackley M, Dolan S, Shafee T. Transcriptomics technologies. PLoS Comput Biol. 2017;13(5):1–23.

5. Wang Z, Gerstein M, Snyder M. RNA-Seq: a revolutionary tool for transcriptomics. Nat Rev | Genet. 2009;VOLUME 10(jANUARy 2009):57–63.

6. Van Vliet AHM. Next generation sequencing of microbial transcriptomes: Challenges and opportunities. FEMS Microbiol Lett. 2010;302(1):1–7.

7. Raz T, Kapranov P, Lipson D, Letovsky S, Milos PM, Thompson JF. Protocol dependence of sequencing-based gene expression measurements. PLoS One. 2011;6(5).

8. Hurd PJ, Nelson CJ. Advantages of next-generation sequencing versus the microarray in epigenetic research. Briefings Funct Genomics Proteomics. 2009;8(3):174–83.

9. Marioni JC, Mason CE, Mane SM, Stephens M, Gilad Y. RNA-seq: An assessment of technical reproducibility and comparison with gene expression arrays. Genome Res. 2008;18(9):1509–17.

10. O'Connell M. Differential expression, class discovery and class prediction using S-PLUS and S+ArrayAnalyzer. ACM SIGKDD Explor Newsl. 2003;5(2):38–47.

11. Slonim DK, Tamayo P, Mesirov JP, Golub TR, Lander ES. Class prediction and discovery using gene expression data. Proc Annu Int Conf Comput Mol Biol RECOMB. 2000;263–72.

12. Hajizadeh N, Zhang M, Akerman M, Kohn N, Mathew A, Hadjiliadis D, et al. Survival models to support shared decision-making about advance care planning for people with advanced stage cystic fibrosis. BMJ Open Respir Res. 2021;8(1):1–14.

13. Alizadeh AA, Elsen MB, Davis RE, Ma CL, Lossos IS, Rosenwald A, et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. Nature. 2000;403(6769):503–11.

14. Rosenberg J, Chia YL, Plevritis S. The effect of age, race, tumor size, tumor grade, and disease stage on invasive ductal breast cancer survival in the U.S. SEER database. Breast Cancer Res Treat. 2005;89(1):47–54.

15. Zhu B, Song N, Shen R, Arora A, Machiela MJ, Song L, et al. Integrating Clinical and

Multiple Omics Data for Prognostic Assessment across Human Cancers. Sci Rep [Internet]. 2017;7(1):1–13. Available from: http://dx.doi.org/10.1038/s41598-017-17031-8

16. Wang C, Machiraju R, Huang K. Breast cancer patient stratification using a molecular regularized consensus clustering method. Methods [Internet]. 2014;67(3):304–12. Available from: http://dx.doi.org/10.1016/j.ymeth.2014.03.005

17. Rafiq S, Tapper W, Collins A, Khan S, Politopoulos I, Gerty S, et al. Identification of inherited genetic variations influencing prognosis in early-onset breast cancer. Cancer Res. 2013;73(6):1883–91.

18. Silveira DRA, Quek L, Santos IS, Corby A, Coelho-Silva JL, Pereira-Martins DA, et al. Integrating clinical features with genetic factors enhances survival prediction for adults with acute myeloid leukemia. Blood Adv. 2020;4(10):2339–50.

19. Klein PJ, Moeschberger ML. SURVIVAL ANALYSIS Techniques for Censored and Truncated Data. Springer; 2003.

20. E.L. Kaplan PM. Nonparametric Estimation from Incomplete Observations Author (s): E . L . Kaplan and Paul Meier Source : Journal of the American Statistical Association , Vol . 53 , No . 282 (Jun ., 1958), pp . 457- Published by : American Statistical Association Sta. Am Stat Assoc. 1958;53(282):457–81.

21. Society RS, Society RS. Asymptotically Efficient Rank Invariant Test Procedures Author (s): Richard Peto and Julian Peto Reviewed work (s): Source : Journal of the Royal Statistical Society . Series A (General), Vol . 135 , No . 2 (1972), pp . Published by : Wiley for th. 2013;135(2):185–207.

22. KLEIN J. Small sample moments of some estimators of the variance of the Kaplan-Meier and Nelson-Aalen estimators. Scand J Stat. 1991;18(4):333–40.

23. Cutler SJ, Ederer F, Bethesd L. MAXIMUM UTILIZATION ANALYZING SURVIVAL OF THE LIFE TABLE METHOD the method and its uses have been admirably described by a number of authors , 2-6. Survival (Lond) [Internet]. 1958; Available from: file:///E:/Todas as coisas da Tali/Leituras/Artigos/Cutler and Ederer 1958.pdf

24. Cox DR. Regression Models and Life-Tables Authors (s): D . R . Cox Source : Journal of the Royal Statistical Society . Series B (Methodological), Vol . 34 , No . 2 Published by : Wiley for the Royal Statistical Society Stable URL : http://www.jstor.org/stable. J R Stat Soc. 1972;34(2):187–220.

25. Simon N, Friedman J, Hastie T, Tibshirani R. Regularization paths for Cox's proportional hazards model via coordinate descent. J Stat Softw. 2011;39(5):1–13.

26. Hesterberg T, Choi NH, Meier L, Fraley C. Least angle and ℓ 1 penalized regression: A review. Stat Surv. 2008;2:61–93.

27. Fisher LD, Lin DY. Time-dependent covariates in the cox proportional-hazards regression model. Annu Rev Public Health. 1999;20(6):145–57.

28. Wang Z, Wang CY. Statistical Applications in Genetics and Molecular Biology Buckley-James Boosting for Survival Analysis with High-Dimensional Biomarker

Data Buckley-James Boosting for Survival Analysis with High-Dimensional Biomarker Data ∗ . Stat Appl Genet Mol Biol. 2010;9(1).

29. Wood SN, Augustin NH. GAMs with integrated model selection using penalized regression splines and applications to environmental modelling. Ecol Modell. 2002;157(2–3):157–77.

30. Schmid M, Hothorn T. Flexible boosting of accelerated failure time models. BMC Bioinformatics. 2008;9:1–13.

31. Witten DM, Tibshirani R. Survival Analysis with high-dimensional covariates. Stat Methods Med Res. 2010;19(1):29–51.

32. Jia M, Yuan DY, Lovelace TC, Hu M, Benos P V. Causal discovery in high-dimensional, multicollinear datasets. Front Epidemiol. 2022;2(5).

33. Ma B, Yan G, Chai B, Hou X. XGBLC: an improved survival prediction model based on XGBoost. Bioinformatics. 2022;38(2):410–8.

34. Tibshirani R. The lasso method for variable selection in the cox model. Stat Med. 1997;16(4):385–95.

35. Hoerl AE, Kennard RW. Ridge Regression: Biased Estimation for Nonorthogonal Problems. Technometrics. 1970;12(1):55–67.

36. Zou H, Hastie T. Regularization and variable selection via the elastic net. J R Stat Soc Ser B Stat Methodol. 2005;67(2):301–20.

37. Lee S, Lim H. Review of statistical methods for survival analysis using genomic data. Genomics and Informatics. 2019;17(4).

38. Breiman LEO. Random Forests. 2001;5–32.

39. Ishwaran H, Kogalur UB, Blackstone EH, Lauer MS. Random survival forests. Ann Appl Stat. 2008;2(3):841–60.

40. Kwak LW, Halpern J, Olshen RA, Horning SJ. Prognostic significance of actual dose intensity in diffuse large-cell lymphoma: Results of a tree-structured survival analysis. J Clin Oncol. 1990;8(6):963–77.

41. Gordon L, Olshen RA. Tree-structured survival analysis. Cancer Treat Rep. 1985;69(10):1065–9.

42. Landoni G, Greco T, Biondi-Zoccai G, Neto CN, Febres D, Pintaudi M, et al. Anaesthetic drugs and survival: A bayesian network meta-analysis of randomized trials in cardiac surgery. Br J Anaesth [Internet]. 2013;111(6):886–96. Available from: http://dx.doi.org/10.1093/bja/aet231

43. Biganzoli E, Boracchi P, Mariani L, Marubini E. Feed forward neural networks for the analysis of censored survival data: A partial logistic regression approach. Stat Med. 1998;17(10):1169–86.

44. Lee YJ, Mangasarian O, Wolberg W. Breast cancer survival and chemotherapy: A support vector machine analysis. 2000;0000(December):1–10.

45. Hothorn T, Lausen B, Benner A, Radespiel-Tröger M. Bagging survival trees. Stat Med. 2004;23(1):77–91.

46. Hothorn T, Bühlmann P, Dudoit S, Molinaro A, Van Der Laan MJ. Survival ensembles. Biostatistics. 2006;7(3):355–73.

47. Faraggi D, Simon R. A neural network model for survival data. Stat Med. 1995;14(1):73–82.

48. Paul R, Hawkins S, Balagurunathan Y, Schabath M, Gillies R, Hall L, et al. Deep Feature Transfer Learning in Combination with Traditional Features Predicts Survival among Patients with Lung Adenocarcinoma. Tomography. 2016;2(4):388–95.

49. Li Y, Wang J, Ye J, Reddy CK. A multi-task learning formulation for survival analysis. Proc ACM SIGKDD Int Conf Knowl Discov Data Min. 2016;13-17-Augu:1715–24.

50. Bøvelstad HM, Nygård S, Borgan Ø. Survival prediction from clinico-genomic models - a comparative study. BMC Bioinformatics. 2009;10:1–9.

51. van Wieringen WN, Kun D, Hampel R, Boulesteix AL. Survival prediction using gene expression data: A review and comparison. Comput Stat Data Anal [Internet]. 2009;53(5):1590–603. Available from: http://dx.doi.org/10.1016/j.csda.2008.05.021

52. Spooner A, Chen E, Sowmya A, Sachdev P, Kochan NA, Trollor J, et al. A comparison of machine learning methods for survival analysis of high-dimensional clinical data for dementia prediction. Sci Rep [Internet]. 2020;10(1):1–10. Available from: https://doi.org/10.1038/s41598-020-77220-w

53. Herrmann M, Probst P, Hornung R, Jurinovic V, Boulesteix AL. Large-scale benchmark study of survival prediction methods using multi-omics data. Brief Bioinform. 2021;22(3):1–15.

54. Ma B, Geng Y, Meng F, Yan G, Song F. Identification of a sixteen-gene prognostic biomarker for lung adenocarcinoma using a machine learning method. J Cancer. 2020;11(5):1288–98.

55. Huang Z, Johnson TS, Han Z, Helm B, Cao S, Zhang C, et al. Deep learning-based cancer survival prognosis from RNA-seq data: Approaches and evaluations. BMC Med Genomics [Internet]. 2020;13(Suppl 5):1–12. Available from: http://dx.doi.org/10.1186/s12920-020-0686-1

56. Ching T, Zhu X, Garmire LX. Cox-nnet: An artificial neural network method for prognosis prediction of high-throughput omics data. PLoS Comput Biol. 2018;14(4):1–18.

57. Wang L. Deep Learning Techniques to Diagnose Lung Cancer. Cancers (Basel). 2022;14(22).

58. Grimes T, Walker AR, Datta S, Datta S. Predicting survival times for neuroblastoma patients using RNA-seq expression profiles. Biol Direct. 2018;13(1):1–15.

59. Jardillier R, Koca D, Chatelain F, Guyon L. Prognosis of lasso-like penalized Cox models with tumor profiling improves prediction over clinical data alone and benefits from bi-dimensional pre-screening. BMC Cancer [Internet]. 2022;22(1):1–16.

Available from: https://doi.org/10.1186/s12885-022-10117-1

60. Ding D, Lang T, Zou D, Tan J, Chen J, Zhou L, et al. Machine learning-based prediction of survival prognosis in cervical cancer. BMC Bioinformatics [Internet]. 2021;22(1):1–17. Available from: https://doi.org/10.1186/s12859-021-04261-x

61. Craig E, Zhong C, Tibshirani R. Survival stacking: casting survival analysis as a classification problem. 2021;1–17. Available from: http://arxiv.org/abs/2107.13480

62. Nguyen T, Bhatti A, Yang S, Nahavandi S. RNA-seq count data modelling by grey relational analysis and nonparametric Gaussian process. PLoS One. 2016;11(10):1–18.

63. Witten DM. Classification and clustering of sequencing data using a poisson model. Ann Appl Stat. 2011;5(4):2493–518.

64. Robinson MD, Smyth GK. Small-sample estimation of negative binomial dispersion, with applications to SAGE data. Biostatistics. 2008;9(2):321–32.

65. Anders S, Huber W. Differential expression analysis for sequence count data. Nat Preced. 2010;

66. Eling N, Richard AC, Richardson S, Marioni JC, Vallejos CA. Correcting the Mean-Variance Dependency for Differential Variability Testing Using Single-Cell RNA Sequencing Data. Cell Syst [Internet]. 2018;7(3):284-294.e12. Available from: https://doi.org/10.1016/j.cels.2018.06.011

67. Zwiener I, Frisch B, Binder H. Transforming RNA-Seq data to improve the performance of prognostic gene signatures. PLoS One. 2014;9(1):1–13.

68. Love MI, Huber W, Anders S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol. 2014;15(12):1–21.

69. Law CW, Chen Y, Shi W, Smyth GK. Voom: Precision weights unlock linear model analysis tools for RNA-seq read counts. Genome Biol. 2014;15(2):1–17.

70. Ritchie ME, Diyagama D, Neilson J, van Laar R, Dobrovic A, Holloway A, et al. Empirical array quality weights in the analysis of microarray data. BMC Bioinformatics. 2006;7.

71. Zararsiz G, Goksuluk D, Klaus B, Korkmaz S, Eldem V, Karabulut E, et al. voomDDA: Discovery of diagnostic biomarkers and classification of RNA-seq data. PeerJ. 2017;2017(10):1–27.

72. Cephe A, Koçhan N, Zararsız GE, Eldem V, Coşgun E, Karabulut E, et al. voomSOM: voom-based Self-Organizing Maps for Clustering RNASequencing Data. Curr Bioinform. 2022;18(2):154–69.

73. Klau S, Jurinovic V, Hornung R, Herold T, Boulesteix AL. Priority-Lasso: A simple hierarchical approach to the prediction of clinical outcome using multi-omics data. BMC Bioinformatics. 2018;19(1):1–14.

74. Boulesteix AL, De Bin R, Jiang X, Fuchs M. IPF-LASSO: Integrative L1-Penalized Regression with Penalty Factors for Prediction Based on Multi-Omics Data. Comput Math Methods Med. 2017;2017.

75. Ferlay J, Colombet M, Soerjomataram I, Parkin DM, Piñeros M, Znaor A, et al. Cancer statistics for the year 2020: An overview. Int J Cancer. 2021;149(4):778–89.

76. Neal RD, Tharmanathan P, France B, Din NU, Cotton S, Fallon-Ferguson J, et al. Is increased time to diagnosis and treatment in symptomatic cancer associated with poorer outcomes? Systematic review. Br J Cancer. 2015;112:S92–107.

77. Kakushadze Z, Raghubanshi R, Yu W. Estimating cost savings from early cancer diagnosis. Data. 2017;2(3):1–16.

78. Kattan MW, Kantoff PW, Nelson JB, Carroll PR, Roach M, Higano CS. Comparison of Cox regression with other methods for determining prediction models and nomograms. J Urol. 2003;170(6):6–10.

79. Singh R, Mukhopadhyay K. Survival analysis in clinical trials: Basics and must know areas. Perspect Clin Res. 2011;2(4):145.

80. Henderson R. Problems and prediction in survival- data analysis. Stat Med. 1995;14(2):161–84.

81. McCann RM. Comfort Care for Terminally III Patients. Jama. 1994;272(16):1263.

82. Waljee AK, Rogers MAM, Lin P, Singal AG, Stein JD, Marks RM, et al. Short term use of oral corticosteroids and related harms among adults in the United States: population based cohort study. BMJ. 2017;357:j1415.

83. Sotiriou C, Wirapati P, Loi S, Harris A, Fox S, Smeds J, et al. Gene expression profiling in breast cancer: Understanding the molecular basis of histologic grade to improve prognosis. J Natl Cancer Inst. 2006;98(4):262–72.

84. Bielinski SJ, Olson JE, Pathak J, Weinshilboum RM, Wang L, Lyke KJ, et al. Preemptive genotyping for personalized medicine: Design of the right drug, right dose, right timedusing genomic data to individualize treatment protocol. Mayo Clin Proc. 2014;89(1):25–33.

85. Williamson R, Anderson W, Duckett S, Frazer I, Hillyard C, Kowal E, et al. The Future of Precision Medicine in Australia. Report for the Australian Council of Learned Academies. 2018. 1–196 p.

86. Ravina B, Tanner C, DiEuliis D, Eberly S, Flagg E, Galpern WR, et al. A longitudinal program for biomarker development in Parkinson's disease: A feasibility study. Mov Disord. 2009;24(14):2081–90.

87. Pennathur A, Farkas A, Krasinskas AM, Ferson PF, Gooding WE, Gibson MK, et al. Esophagectomy for T1 Esophageal Cancer: Outcomes in 100 Patients and Implications for Endoscopic Therapy. Ann Thorac Surg [Internet]. 2009;87(4):1048–55. Available from: http://dx.doi.org/10.1016/j.athoracsur.2008.12.060

88. Behjati S, Tarpey PS. What is next generation sequencing? Arch Dis Child Educ Pract Ed. 2013;98(6):236–8.

89. Serratì S, de Summa S, Pilato B, Petriella D, Lacalamita R, Tommasi S, et al. Next-generation sequencing: Advances and applications in cancer diagnosis. Onco Targets Ther. 2016;9:7355–65.

90. Hess JF, Kohl TA, Kotrová M, Rönsch K, Paprotka T, Mohr V, et al. Library preparation for next generation sequencing: A review of automation strategies. Biotechnol Adv [Internet]. 2020;41(March):107537. Available from: https://doi.org/10.1016/j.biotechadv.2020.107537

91. Illumina Inc. Illumina sequencing introduction. Illumina Seq Introd [Internet]. 2017;(October):1–8. Available from: https://www.illumina.com/documents/products/illumina_sequencing_introduction.pdf

92. Zararsiz G. Development and application of novel machine learning approaches for rna-seq data classification. 2015.

93. Bentley DR, Balasubramanian S, Swerdlow HP, Smith GP, Milton J, Brown CG, et al. Accurate whole human genome sequencing using reversible terminator chemistry. Nature. 2008;456(7218):53–9.

94. Kim D, Pertea G, Trapnell C, Pimentel H, Kelley R, Salzberg SL. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. Genome Biol. 2013;14(R36).

95. Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, et al. STAR: Ultrafast universal RNA-seq aligner. Bioinformatics. 2013;29(1):15–21.

96. Liao Y, Smyth GK, Shi W. FeatureCounts: An efficient general purpose program for assigning sequence reads to genomic features. Bioinformatics. 2014;30(7):923–30.

97. Robinson MD, McCarthy DJ, Smyth GK. edgeR: A Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics. 2009;26(1):139–40.

98. Rao MS, Van Vleet TR, Ciurlionis R, Buck WR, Mittelstadt SW, Blomme EAG, et al. Comparison of RNA-Seq and microarray gene expression platforms for the toxicogenomic evaluation of liver from short-term rat toxicity studies. Front Genet. 2019;10(JAN):1–16.

99. Zhao S, Fung-Leung WP, Bittner A, Ngo K, Liu X. Comparison of RNA-Seq and microarray in transcriptome profiling of activated T cells. PLoS One. 2014;9(1).

100. Cock PJA, Fields CJ, Goto N, Heuer ML, Rice PM. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Nucleic Acids Res. 2009;38(6):1767–71.

101. S A. FastQC: a quality control tool for high throughput sequence data [Internet]. 2010. Available from: http://www.bioinformatics.babraham.ac.uk/projects/fastqc

102. Jiang H, Lei R, Ding SW, Zhu S. Skewer: A fast and accurate adapter trimmer for next-generation sequencing paired-end reads. BMC Bioinformatics. 2014;15(1):1–12.

103. Conesa A, Madrigal P, Tarazona S, Gomez-Cabrero D, Cervera A, McPherson A, et al. Erratum: A survey of best practices for RNA-seq data analysis [Genome Biol. (2016), 17, 13] doi: 10.1186/s13059-016-0881-8. Genome Biol. 2016;17(1):16–7.

104. Patel RK, Jain M. NGS QC toolkit: A toolkit for quality control of next generation sequencing data. PLoS One. 2012;7(2).

105. Deluca DS, Levin JZ, Sivachenko A, Fennell T, Nazaire MD, Williams C, et al. RNA-SeQC: RNA-seq metrics for quality control and process optimization. Bioinformatics. 2012;28(11):1530–2.

106. Bolger AM, Lohse M, Usadel B. Trimmomatic: A flexible trimmer for Illumina sequence data. Bioinformatics. 2014;30(15):2114–20.

107. Schmieder R, Edwards R. Quality control and preprocessing of metagenomic datasets. Bioinformatics. 2011;27(6):863–4.

108. Chen Y, Chen Y, Shi C, Huang Z, Zhang Y, Li S, et al. SOAPnuke: A MapReduce acceleration-supported software for integrated quality control and preprocessing of high-throughput sequencing data. Gigascience. 2018;7(1):1–6.

109. Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol. 2009;10(3).

110. Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics. 2009;25(14):1754–60.

111. Kim D, Langmead B, Salzberg SL. HISAT: A fast spliced aligner with low memory requirements. Nat Methods. 2015;12(4):357–60.

112. Trapnell C, Williams BA, Pertea G, Mortazavi A, Kwan G, Van Baren MJ, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. Nat Biotechnol. 2010;28(5):511–5.

113. Pertea M, Pertea GM, Antonescu CM, Chang TC, Mendell JT, Salzberg SL. StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. Nat Biotechnol. 2015;33(3):290–5.

114. Grabherr MG, Haas BJ, Yassour M, Levin JZ, Thompson DA, Amit I, et al. Full-length transcriptome assembly from RNA-Seq data without a reference genome. Nat Biotechnol. 2011;29(7):644–52.

115. Hurgobin B. Short Read Alignment Using SOAP2. In: Plant Bioinformatics. Humana Press, New York, NY; 2016. p. 241–252.

116. Robertson G, Schein J, Chiu R, Corbett R, Field M, Jackman SD, et al. De novo assembly and analysis of RNA-seq data. Nat Methods. 2010;7(11):909–12.

117. Li B, Dewey CN. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. BMC Bioinformatics. 2011;12(323).

118. Bray NL, Pimentel H, Melsted P, Pachter L. Near-optimal probabilistic RNA-seq quantification. Nat Biotechnol. 2016;34(5):525–7.

119. Patro R, Duggal G, Love MI, Irizarry RA, Kingsford C. Salmon provides fast and bias-aware quantification of transcript expression. Nat Methods. 2017;14(4):417–9.

120. Anders S, Pyl PT, Huber W. HTSeq-A Python framework to work with high-throughput sequencing data. Bioinformatics. 2015;31(2):166–9.

121. Roberts A, Pachter L. Streaming fragment assignment for real-time analysis of

sequencing experiments. Nat Methods. 2013;10(1):71–3.

122. Anders S, Reyes A, Huber W. Detecting differential usage of exons from RNA-seq data. Genome Res. 2012;22(10):2008–17.

123. Patro R, Mount SM, Kingsford C. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. Nat Biotechnol. 2014;32(5):462–4.

124. Chen HM, MacDonald JA. Network analysis of TCGA and GTEx gene expression datasets for identification of trait-associated biomarkers in human cancer. STAR Protoc [Internet]. 2022;3(1):101168. Available from: https://doi.org/10.1016/j.xpro.2022.101168

125. Chen HM, MacDonald JA. Network analysis of TCGA and GTEx gene expression datasets for identification of trait-associated biomarkers in human cancer. STAR Protoc. 2022 Mar 18;3(1).

126. Law CW, Alhamdoosh M, Su S, Dong X, Tian L, Smyth GK, et al. RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR. F1000Research. 2018;5(1):1–29.

127. Park G, Park JK, Shin SH, Jeon HJ, Kim NKD, Kim YJ, et al. Characterization of background noise in capture-based targeted sequencing data. Genome Biol. 2017;18(1):1–13.

128. Soneson C, Gerster S, Delorenzi M. Batch effect confounding leads to strong bias in performance estimates obtained by cross-validation. PLoS One. 2014;9(6):1–12.

129. Bullard JH, Purdom E, Hansen KD, Dudoit S. Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. 2010;

130. Oshlack A, Wakefield MJ. Transcript length bias in RNA-seq data confounds systems biology. Biol Direct. 2009;4:1–10.

131. Risso D, Schwartz K, Sherlock G, Dudoit S. GC-Content Normalization for RNA-Seq Data. BMC Bioinformatics. 2011;

132. Robinson MD, Oshlack A. A scaling normalization method for differential expression analysis of RNA-seq data. Genome Biol [Internet]. 2010;11(3):1–9. Available from: http://genomebiology.com/2010/11/3/R25

133. Dillies MA, Rau A, Aubert J, Hennequet-Antier C, Jeanmougin M, Servant N, et al. A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis. Brief Bioinform. 2013;14(6):671–83.

134. Bolstad BM, Irizarry RA, Åstrand M, Speed TP. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. Bioinformatics. 2003;19(2):185–93.

135. Han H, Men K. How does normalization impact RNA-seq disease diagnosis? J Biomed Inform [Internet]. 2018;85(July):80–92. Available from: https://doi.org/10.1016/j.jbi.2018.07.016

136. Mortazavi A, Williams BA, McCue K, Schaeffer L, Wold B. Mapping and quantifying

mammalian transcriptomes by RNA-Seq. Nat Methods. 2008;5(7):621–8.

137. Wagner P, Kin K, Lynch VJ. Measurement of mRNA abundance using RNA-seq data : RPKM measure is inconsistent among samples. 2012;281–5.

138. Trapnell C, Hendrickson DG, Sauvageau M, Goff L, Rinn JL, Pachter L. Differential analysis of gene regulation at transcript resolution with RNA-seq. Nat Biotechnol. 2013;31(1):46–53.

139. Maza E, Frasse P, Senin P, Bouzayen M, Zouine M. Comparison of normalization methods for differential gene expression analysis in RNA-Seq experiments: A matter of relative size of studied transcriptomes. Commun Integr Biol. 2013;6(6).

140. Zhang Z, Yu D, Seo M, Hersh CP, Weiss ST, Qiu W. Novel Data Transformations for RNA-seq Differential Expression Analysis. Sci Rep. 2019;9(1):1–12.

141. Ahlmann-Eltze C, Huber W. Comparison of Transformations for Single-Cell RNA-Seq Data. bioRxiv [Internet]. 2022;20(May):2021.06.24.449781. Available from: http://biorxiv.org/content/early/2022/11/12/2021.06.24.449781.abstract

142. Agresti A. 3Rd-Ed-Alan_Agresti_Categorical_Data_Analysis.Pdf. Vol. 47, International encyclopedia of statistical science. 2013. p. 755–8.

143. Cloonan N, Forrest ARR, Kolle G, Gardiner BBA, Faulkner GJ, Brown MK, et al. Stem cell transcriptome profiling via massive-scale mRNA sequencing. Nat Methods. 2008;5(7):613–9.

144. Perkins TT, Kingsley RA, Fookes MC, Gardner PP, James KD, Yu L, et al. A strand-specific RNA-seq analysis of the transcriptome of the typhoid bacillus Salmonella typhi. PLoS Genet. 2009;5(7).

145. Parikh A, Miranda ER, Katoh-Kurasawa M, Fuller D, Rot G, Zagar L, et al. Conserved developmental transcriptomes in evolutionarily divergent species. Genome Biol. 2010;11(3).

146. Zhou YH, Xia K, Wright FA. A powerful and flexible approach to the analysis of RNA sequence count data. Bioinformatics. 2011;27(19):2672–8.

147. Srivastava S, Chen L. A two-parameter generalized Poisson model to improve the analysis of RNA-seq data. Nucleic Acids Res. 2010;38(17):1–15.

148. McCarthy DJ, Chen Y, Smyth GK. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. Nucleic Acids Res. 2012;40(10):4288–97.

149. Smyth GK. Linear models and empirical bayes methods for assessing differential expression in microarray experiments. Stat Appl Genet Mol Biol. 2004;3(1):1–26.

150. Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, et al. Limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Res. 2015;43(7):e47.

151. Liu R, Holik AZ, Su S, Jansz N, Chen K, Leong HS, et al. Why weight? Modelling sample and observational level variability improves power in RNA-seq analyses.

Nucleic Acids Res. 2015;43(15).

152. Kursa MB, Rudnicki WR. Feature selection with the boruta package. J Stat Softw. 2010;36(11):1–13.

153. Guo P, Luo Y, Mai G, Zhang M, Wang G, Zhao M, et al. Gene expression profile based classification models of psoriasis. Genomics. 2014;103(1):48–55.

154. Saulnier DM, Riehle K, Mistretta TA, Diaz MA, Mandal D, Raza S, et al. Gastrointestinal microbiome signatures of pediatric patients with irritable bowel syndrome. Gastroenterology [Internet]. 2011;141(5):1782–91. Available from: http://dx.doi.org/10.1053/j.gastro.2011.06.072

155. Degenhardt F, Seifert S, Szymczak S. Evaluation of variable selection methods for random forests and omics data sets. Brief Bioinform. 2019;20(2):492–503.

156. Kvamme H, Borgan O, Scheel I. Time-to-event prediction with neural networks and cox regression. J Mach Learn Res. 2019;20:1–30.

157. Kleinbaum DG, Mitchel K. Survival Analysis: A Self-Learning Text. Vol. 39, Technometrics. 1997. 228–229 p.

158. Lee ET, Wang JW. Statistical Methods for Survival Data Analysis. John Wiley & Sons, Inc.; 2003. 243–255 p.

159. Tobin BYJ. Estimation of Relationships for Limited Dependent Variables Author (s): James Tobin Published by : The Econometric Society Stable URL : http://www.jstor.org/stable/1907382 . OF RELATIONSHIPS FOR LIMITED DEPENDENT VARIABLES ’. Econometrica. 1985;26(1):24–36.

160. Buckley J, James I. Linear regression with censored data. Biometrika. 1979;66(3):429–36.

161. Wang S, Nan B, Zhu J, Beer DG. Doubly penalized Buckley-James method for survival data with high-dimensional covariates. Biometrics. 2008;64(1):132–40.

162. Kyung M, Gilly J, Ghoshz M, Casellax G. Penalized regression, standard errors, and Bayesian lassos. Bayesian Anal. 2010;5(2):369–412.

163. Li Y, Vinzamuri B, Reddy CK. Regularized weighted linear regression for high-dimensional censored data. 16th SIAM Int Conf Data Min 2016, SDM 2016. 2016;1(c):45–53.

164. Bach F, Jenatton R, Mairal J, Obozinski G. Structured sparsity through convex optimization. Stat Sci. 2012;27(4):450–68.

165. Kalbfleisch JD, Prentice RL. The Statistical Analysis of Failure Time Data. John Wiley & Sons, Inc. All rights reserved.; 2002.

166. Deo SV, Deo V, Sundaram V. Survival analysis—part 2: Cox proportional hazards model. Indian J Thorac Cardiovasc Surg. 2021;37(2):229–33.

167. van Belle V, Pelckmans K, van Huffel S, Suykens JAK. Improved performance on high-dimensional survival data by application of survival-SVM. Bioinformatics.

2011;27(1):87–94.

168. Radespiel-Tröger M, Rabenstein T, Schneider HT, Lausen B. Comparison of tree-based methods for prognostic stratification of survival data. Artif Intell Med. 2003;28(3):323–41.

169. Verweij PJM, Van Houwelingen HC. Penalized likelihood in Cox regression. Stat Med. 1994;13:2427–36.

170. Shrinkage R. Regression Shrinkage and Selection via the Lasso Author (s): Robert Tibshirani Source : Journal of the Royal Statistical Society . Series B (Methodological), Vol . 58 , No . 1 (1996), Published by : Wiley for the Royal Statistical Society Stable URL. 2016;58(1):267–88.

171. Gui J, Li H. Penalized Cox regression analysis in the high-dimensional and low-sample size settings, with applications to microarray gene expression data. Bioinformatics. 2005;21(13):3001–8.

172. Ternès N, Rotolo F, Michiels S. Empirical extensions of the lasso penalty to reduce the false discovery rate in high-dimensional Cox regression models. Stat Med. 2016;35(15):2561–73.

173. Binder H, Schumacher M. Allowing for mandatory covariates in boosting estimation of sparse high-dimensional survival models. BMC Bioinformatics. 2008;9:1–10.

174. Breiman L. Classification and Regression Trees. Taylor & Francis Group; 1984.

175. Mingers J. An empirical comparison of selection measures for decision-tree induction. Mach Learn. 1989;3(4):319–42.

176. Bou-Hamad I, Larocque D, Ben-Ameur H. A review of survival trees. Stat Surv. 2011;5(2011):44–71.

177. Quinlan JR. Induction of decision trees. Mach Learn. 1986;1(1):81–106.

178. Ciampi A, Negassa A, Lou Z. Tree-structured prediction for censored survival data and the cox model. J Clin Epidemiol. 1995;48(5):675–89.

179. Marubini E, Morabito A, Valsecchi MG. Prognostic factors and risk groups: Some results given by using an algorithm suitable for censored survival data. Stat Med. 1983;2(2):295–303.

180. Dietterich TG. Ensemble methods in machine learning. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics). 2000;1857 LNCS:1–15.

181. Ciampi A, Thiffault J, Nakache JP, Asselain B. Stratification by stepwise regression, correspondence analysis and recursive partition. Comput Stat Data Anal. 1986;4:185–204.

182. Segal MR. Regression Trees for Censored Data. 1988;44(1):35–47.

183. Jefrey H. Butler, Elzabeth A. Gilpin LG and RAO. Tree-structured survival analysis, II. 1989.

184. Davis RB, Anderson JR. Exponential survival trees. Stat Med. 1989;8(8):947–61.

185. Therneau TM, Grambsch PM, Fleming TR. Biometrika Trust Martingale-Based Residuals for Survival Models Published by : Oxford University Press on behalf of Biometrika Trust Stable URL : http://www.jstor.org/stable/2336057 REFERENCES Linked references are available on JSTOR for this article : Yo. 2016;77(1):147–60.

186. LeBlanc M, Crowley J. Relative Risk Trees for Censored Survival Data. Biometrics. 1992;48(2):411–25.

187. Kele S, Segal MR. Residual-based tree-structured survival analysis. Stat Med. 2002;21(2):313–26.

188. LeBlanc M, Crowley J. Survival trees by goodness of split. J Am Stat Assoc. 1993;88(422):457–67.

189. Intrator O, Kooperberg C. Trees and splines in survival analysis. Stat Methods Med Res. 1995;4(3):237–61.

190. Zhang HP, Singer B. Recursive partitioning in the health sciences. New York, NY: Springer; 1999.

191. Breiman L. Software for the masses. 2002.

192. Molinaro AM, Dudoit S, Van Der Laan MJ. Tree-based multivariate regression and density estimation with right-censored data. J Multivar Anal. 2004;90(1 SPEC. ISS.):154–77.

193. Jin H, Lu Y, Stone K, Black DM. Alternative tree-structured survival analysis based on variance of survival time. Med Decis Mak. 2004;24(6):670–80.

194. Hothorn T, Hornik K, Zeileis A. Unbiased recursive partitioning: A conditional inference framework. J Comput Graph Stat. 2006;15(3):651–74.

195. Breiman L. Bagging predictors. Mach Learn. 1996;24:123–40.

196. Wang H, Li G. A Selective Review on Random Survival Forests for High Dimensional Data. Quant Biosci. 2017;36(2):85–96.

197. Freund Y, Schapire RE. Experiments with a New Boosting Algorithm. Proc 13th Int Conf Mach Learn [Internet]. 1996;148–156. Available from: http://www.public.asu.edu/~jye02/CLASSES/Fall-2005/PAPERS/boosting-icml.pdf

198. Chen T, Guestrin C. XGBoost : A Scalable Tree Boosting System. 2016;785–94.

199. Friedman JH. Greedy function approximation: A gradient boosting machine. Ann Stat. 2001;29(5):1189–232.

200. Freund Y. An adaptive version of the boost by majority algorithm. Mach Learn. 2001;43(3):293–318.

201. Cortes C, Vapnik V. Support-Vector Networks. Mach Learn. 1995;20:273–97.

202. Shivaswamy PK, Chu W, Jansche M. A support vector approach to censored targets.

Proc - IEEE Int Conf Data Mining, ICDM. 2007;655–60.

203. Van Belle V, Pelckmans K, Van Huffel S, Suykens JAK. Support vector methods for survival analysis: A comparison between ranking and regression approaches. Artif Intell Med [Internet]. 2011;53(2):107–18. Available from: http://dx.doi.org/10.1016/j.artmed.2011.06.006

204. Evers L, Messow CM. Sparse kernel methods for high-dimensional survival data. Bioinformatics. 2008;24(14):1632–8.

205. Statistical Analysis - 2011 - Ishwaran - Random survival forests for high- dimensional data.pdf.

206. D'Agostino RB, Lee M - L, Belanger AJ, Cupples LA, Anderson K, Kannel WB. Relation of pooled logistic regression to time dependent cox regression analysis: The framingham heart study. Stat Med. 1990;9(12):1501–15.

207. Oehlert GW. A Note on the Delta Method Author (s): Gary W . Oehlert Source : The American Statistician , Feb ., 1992 , Vol . 46 , No . 1 (Feb ., 1992), pp . 27-29 Published by : Taylor & Francis , Ltd . on behalf of the American Statistical Association Stable URL. 1992;46(1):27–9.

208. Cleveland WS. Robust Locally Weighted Regression and Smoothing Scatterplots. J Am Stat Assoc. 1979;74(368):829.

209. Colaprico A, Silva TC, Olsen C, Garofano L, Cava C, Garolini D, et al. TCGAbiolinks: An R/Bioconductor package for integrative analysis of TCGA data. Nucleic Acids Res. 2016;44(8):e71.

210. Liu J, Lichtenberg T, Hoadley KA, Poisson LM, Lazar AJ, Cherniack AD, et al. An Integrated TCGA Pan-Cancer Clinical Data Resource to Drive High-Quality Survival Outcome Analytics. Cell [Internet]. 2018;173(2):400–16. Available from: https://doi.org/10.1016/

211. Lang M, Binder M, Richter J, Schratz P, Pfisterer F, Coors S, et al. mlr3: A modern object-oriented machine learning framework in R. J Open Source Softw. 2019;4(44):1903.

212. Chen Y, Lun ATL, Smyth GK. From reads to genes to pathways: Differential expression analysis of RNA-Seq experiments using Rsubread and the edgeR quasi-likelihood pipeline. F1000Research. 2016;5:1–51.

213. Smyth GK. limma: Linear Models for Microarray Data. Bioinforma Comput Biol Solut Using R Bioconductor. 2005;(2005):397–420.

214. Becker M, Schratz P, Lang M, Bischl B. Package 'mlr3fselect' [Internet]. 2024. Available from: http://ftp2.de.freebsd.org/pub/misc/cran/web/packages/mlr3fselect/mlr3fselect.pdf

215. Sonabend R, Lang M, Kira FJ, Bender A, Bischl B. Data and text mining mlr3proba : an R package for machine learning in survival analysis. 2021;37(February):2789–91.

216. Laj Spytek M, Krzyzi´nski MK, Langbein SH, Baniecki H, Wright MN, Law Biecek P. survex: an R package for explaining machine learning survival models. 2023;1–4.

Available from: https://arxiv.org/abs/2308.16113v1

217. Pihur V, Datta S, Datta S. RankAggreg, an R package for weighted rank aggregation. BMC Bioinformatics. 2009;10:1–10.

218. Harrell FE, Califf RM, Pryor DB, Lee KL, Rosati RA. Evaluating the yield of medical tests. Jama. 1982;247(18):2543–6.

219. Graf E, Schmoor C, Sauerbrei W, Schumacher M. Assessment and comparison of prognostic classification schemes for survival data. Stat Med. 1999;18(17–18):2529–45.

220. mlr3extralearners [Internet]. Available from: https://github.com/mlr-org/mlr3extralearners

221. Boosting TM based, Matrix I. Package ‘ mboost ’ [Internet]. 2023. Available from: https://cran.r-project.org/web/packages/mboost/index.html

222. Hans A, Borchers W, Borchers MHW. Package ‘ pracma .’ 2023; Available from: https://cran.r-project.org/web/packages/pracma/index.html

223. Bühlmann P, Yu B. Boosting With the L 2 Loss Regression and Classification Boosting With the L 2 Loss : Regression and Classi cation. 2011;1459.

224. Date RP, Xml S, Lazydata F, Gpl- L, Hothorn AT, Seibold H, et al. Package ‘ partykit ’ [Internet]. 2023. Available from: https://cran.r-project.org/web/packages/partykit/index.html

225. Graham N. Package ‘ sandwich .’ 2023; Available from: https://cran.r-project.org/web/packages/sandwich/index.html

226. Conditional T, Procedures I, Test P, Description F, Utf- E, Gpl- L, et al. Package ‘ coin ’ [Internet]. 2023. Available from: https://cran.r-project.org/web/packages/coin/index.html

227. Binder H, Allignol A, Schumacher M, Beyersmann J. Boosting for high-dimensional time-to-event data with competing risks. 2009;25(7):890–6.

228. Generalized T, Regression B, Adaboost S. Package ‘ gbm ’ [Internet]. 2024. Available from: https://cran.r-project.org/web/packages/gbm/index.html

229. Friedman JH. Stochastic gradient boosting. Comput Stat Data Anal. 2002;38(4):367–78.

230. Jerome A, Hastie T, Tibshirani R, Tay K, Simon N, Yang J, et al. Package ‘ glmnet ’ R topics documented : 2023; Available from: https://cran.r-project.org/web/packages/glmnet/index.html

231. Friedman J, Hastie T, Tibshirani R. Regularization Paths for Generalized Linear Models via Coordinate Descent. J Stat Softw. 2010;33(1):1–20.

232. Rcpp I, Rcpp L. Package ‘ obliqueRSF .’ 2022; Available from: https://cran.r-project.org/web/packages/obliqueRSF/index.html

233. Orsf T, Cox R, Study JH, Cvd A, Risk PC, November R, et al. Oblique random survival forests 1. 2019;13(3):1847–83.

234. Goeman AJ, Meijer R, Chaturvedi N, Lueder M, Rcpp I, Rcpp L. Package ‘ penalized .’ 2022;1. Available from: https://cran.r-project.org/web/packages/penalized/index.html

235. Dicekriging S. Package ‘mlr3learners’ [Internet]. 2024. Available from: https://cran.r-project.org/web/packages/mlr3learners/index.html

236. Marvin A, Wright N, Wager S, Probst P, Wright MMN. Package ‘ ranger .’ 2023; Available from: https://cran.r-project.org/web/packages/ranger/index.html

237. Fast T, Random U, Ishwaran AH, Kogalur UB, Kogalur MUB, Suggests D, et al. Package ‘ randomForestSRC .’ 2023; Available from: https://cran.r-project.org/web/packages/randomForestSRC/index.html

238. Partitioning TR, Trees R. Package ‘ rpart .’ 2023; Available from: https://cran.r-project.org/web/packages/rpart/index.html

239. Sonabend R, Király FJ. distr6 : R6 Object-Oriented Probability Distributions Interface in R. 2021;13(June):470–92.

240. Lumley T, S- R, Elizabeth A, Cynthia C, Therneau MTM. Package ‘ survival .’ 2024; Available from: https://cran.r-project.org/web/packages/sandwich/index.html

241. Package T, Survival T, Vector S, Fouodo ACJK. Package ‘ survivalsvm .’ 2022; Available from: https://cran.r-project.org/web/packages/survivalsvm/index.html

242. Chen K, Mitchell R, Cano I, Lin M. Package ‘ xgboost ’ R topics documented : 2024; Available from: https://cran.r-project.org/web/packages/xgboost/index.html

243. Install TE. Package ‘mlr3verse.’ 2023;1–4. Available from: https://cran.r-project.org/web/packages/mlr3verse/index.html

244. Spaces TS, Collection D, Lgpl- L, Utf- E, Becker AM, Lang M, et al. Package ‘mlr3tuningspaces.’ 2024; Available from: https://cran.r-project.org/web/packages/mlr3tuningspaces/index.html

245. Vickers AJ, Cronin AM. Traditional Statistical Methods for Evaluating Prediction Models Are Uninformative as to Clinical Value: Towards a Decision Analytic Framework. Semin Oncol [Internet]. 2010;37(1):31–8. Available from: http://dx.doi.org/10.1053/j.seminoncol.2009.12.004

246. Cook NR. Use and misuse of the receiver operating characteristic curve in risk prediction. Circulation. 2007;115(7):928–35.

247. Halligan S, Altman DG, Mallett S. Disadvantages of using the area under the receiver operating characteristic curve to assess imaging tests: A discussion and proposal for an alternative approach. Eur Radiol. 2015;25(4):932–9.

248. Karovic H. Comparison of Pre-processing Methods and Various Machine Learning Models for Survival Analysis on Cancer Data.

249. Tizi W, Berrado A. Machine learning for survival analysis in cancer research: A comparative study. Sci African [Internet]. 2023;21(August):e01880. Available from: https://doi.org/10.1016/j.sciaf.2023.e01880

250. Xiao J, Mo M, Wang Z, Zhou C, Shen J, Yuan J, et al. The Application and Comparison of Machine Learning Models for the Prediction of Breast Cancer Prognosis: Retrospective Cohort Study. JMIR Med Informatics. 2022;10(2):1–11.

251. Hadanny A, Shouval R, Wu J, X CPG, Unger R, Zahger D, et al. This is a repository copy of Machine learning-based prediction of 1-year mortality for acute coronary syndrome . White Rose Research Online URL for this paper : Version : Published Version Article : Hadanny , A , Shouval , R , Wu , J orcid . org / 0000-00. 2022;

252. Hao Y, Liang D, Zhang S, Wu S, Li D, Wang Y, et al. Machine learning for predicting the survival in osteosarcoma patients: Analysis based on American and Hebei Province cohort. Biomol Biomed. 2023;23(5):883–93.

253. Jin X, Sun Y, Zhou T, Leng Y, Guan S, Zhang K, et al. Machine Learning and Prediction of All-Cause Mortality among Chinese Older Adults. 2021;

254. Haynatzki R. Prediction of survival models: A comparison between machine learning and cox regression. AIP Conf Proc. 2022;2522(September).

255. Moncada-Torres A, van Maaren MC, Hendriks MP, Siesling S, Geleijnse G. Explainable machine learning can outperform Cox regression predictions and provide insights in breast cancer survival. Sci Rep [Internet]. 2021;11(1):1–13. Available from: https://doi.org/10.1038/s41598-021-86327-7

256. Linden T, Hanses F, Domingo-Fernández D, DeLong LN, Kodamullil AT, Schneider J, et al. Machine Learning Based Prediction of COVID-19 Mortality Suggests Repositioning of Anticancer Drug for Treating Severe Cases. Artif Intell Life Sci [Internet]. 2021;1(November):100020. Available from: https://doi.org/10.1016/j.ailsci.2021.100020

257. Lynch CM, Abdollahi B, Fuqua JD, de Carlo AR, Bartholomai JA, Balgemann RN, et al. Prediction of lung cancer patient survival via supervised machine learning classification techniques. Int J Med Inform [Internet]. 2017;108(April 2016):1–8. Available from: http://dx.doi.org/10.1016/j.ijmedinf.2017.09.013

258. Kim H, Park T, Jang J, Lee S. Comparison of survival prediction models for pancreatic cancer: Cox model versus machine learning models. Genomics and Informatics. 2022;20(2):1–9.

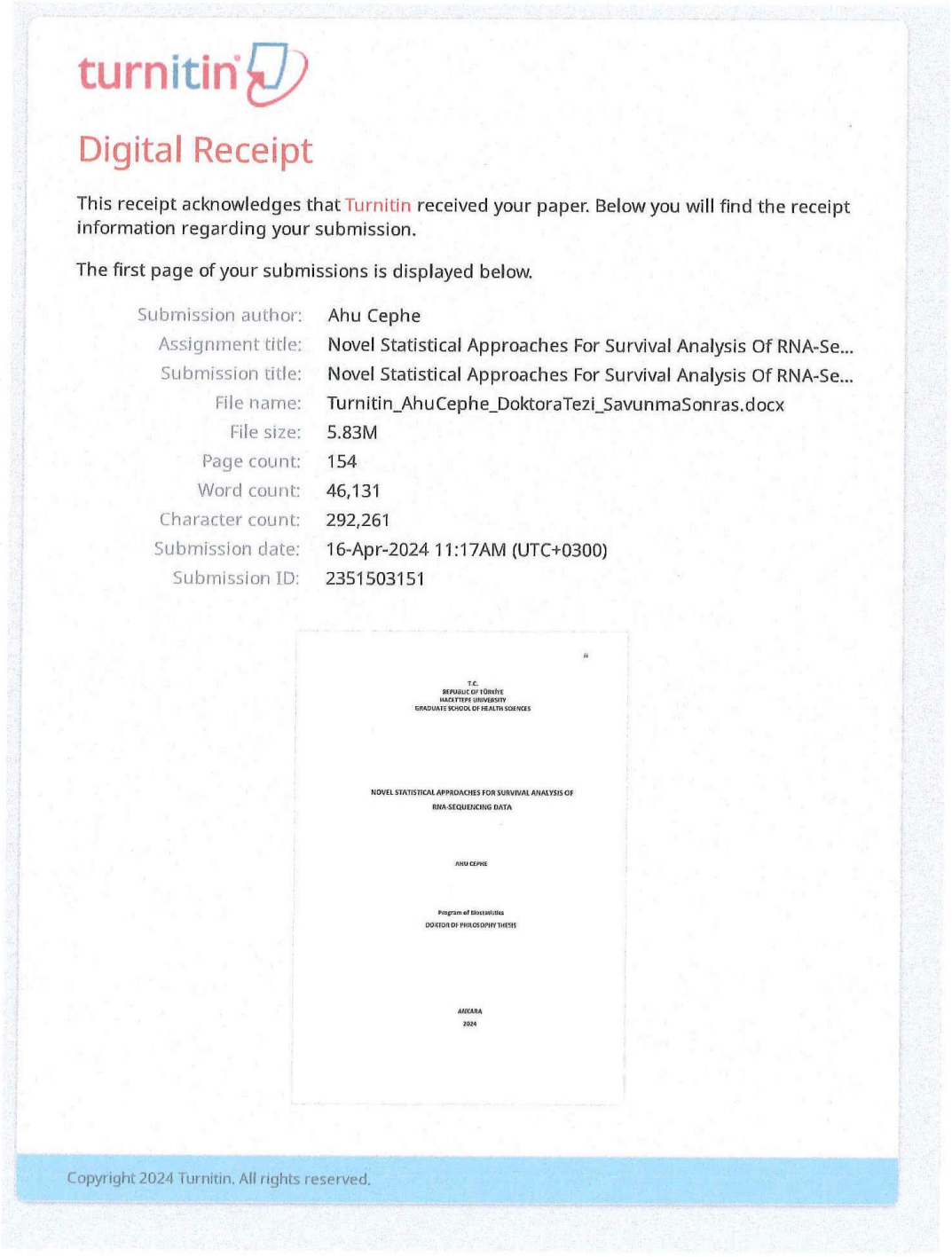
8. APPENDICES

Appendix 1: The codes for analysis and MLSeqSurv R Package, the selected features for models.

The codes for analysis and selected features for models are available at https://github.com/gokmenzararsiz/voomStackLasso.

The codes for MLSeqSurv R Package is available at https://github.com/gokmenzararsiz/MLSeqSurv.

Appendix 2: The Originality Report of Thesis Study.



turnitin

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Ahu Cephe
Assignment title:	Novel Statistical Approaches For Survival Analysis Of RNA-Se...
Submission title:	Novel Statistical Approaches For Survival Analysis Of RNA-Se...
File name:	Turnitin_AhuCephe_DoktoraTezi_SavunmaSonras.docx
File size:	5.83M
Page count:	154
Word count:	46,131
Character count:	292,261
Submission date:	16-Apr-2024 11:17AM (UTC+0300)
Submission ID:	2351503151

T.C.
REPUBLIC OF TÜRKİYE
HACETTEPE UNIVERSITY
GRADUATE SCHOOL OF HEALTH SCIENCES

NOVEL STATISTICAL APPROACHES FOR SURVIVAL ANALYSIS OF RNA-SEQUENCING DATA

AHU CEPHE

Program of Biostatistics
DOKTORA DOCTOR OF PHILOSOPHY THESIS

ANKARA
2024

Novel Statistical Approaches For Survival Analysis Of RNA-Sequencing Data

ORIGINALITY REPORT

9%	5%	7%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.openaccess.hacettepe.edu.tr:8080 Internet Source	1%
2	genomebiology.biomedcentral.com Internet Source	<1%
3	Iulii Vasilev, Mikhail Petrovskiy, Igor Mashechkin. "Chapter 5 Adaptive Sampling forWeighted Log-Rank Survival Trees Boosting", Springer Science and Business Media LLC, 2023 Publication	<1%
4	link.springer.com Internet Source	<1%
5	Ciaran Evans, Johanna Hardin, Daniel M Stoebel. "Selecting between-sample RNA-Seq normalization methods from the perspective of their assumptions", Briefings in Bioinformatics, 2018 Publication	<1%

9. CURRICULUM VITAE (CV)

Ahu Cephe

ORCID: 0000-0001-9374-4495
ResearcherID: GMW-4300-2022
ScopusID: 57464958400

Education

- **Doctor of Philosophy in Biostatistics**
  Hacettepe University — *Ankara/Turkey*
  Thesis: Novel Statistical Approaches For Survival Analysis of RNA-Sequencing Data — *2019 - ...*
- **Master of Science in Biostatistics**
  Erciyes University — *Kayseri/Turkey*
  Thesis: Novel Statistical Approaches in Clustering RNA-Sequencing Data — *2016 - 2019*
- **Bacholar of Science in Statistics**
  Hacettepe University — *Ankara/Turkey*
  Faculty of Science - Statistics — *2005 - 2010*

Experience

- **Lecturer - Erciyes University** — Kayseri/Turkey
  Dean of Research / Rectorate — *Jan 2021 - ...*
  Institutional Data Management and Analytics Unit - Vice Coordinator
- **Lecturer - Hacettepe University** — Ankara/Turkey
  Research Performance Evaluation Office / Rectorate — *Jan 2020 - Dec 2020*
- **Software Specialist and Database Analyst - Şahin Software** — Kayseri/Turkey
  ORACLE — *Jan 2011 - Dec 2015*

Publications

- ***Article - SCI-Expanded***: Uslu K., Ozcelik F., Zararsiz G., Eldem V., **Cephe A.**, Sahin I. O., Yuksel R. C., Sipahioglu H., Ozer Simsek Z., Baspinar O., Akalin H., Simsek Y., Gundogan K., Tutar N., Karayol Akin A., Ozkul Y., Yildiz O., Dundar M. (2023). Deciphering the host genetic factors conferring susceptibility to severe COVID-19 using exome sequencing, GENES AND IMMUNITY, doi:10.1038/s41435-023-00232-9
- ***Article - SCI-Expanded***: Bolat S., Ertürk Zararsız G., Doğan K., Koçhan N., Yerlitaş S. İ., **Cephe A.**, Zararsız G., Cicero A. F. G. (2023). Concordance of LDL-C Estimating Equations with Direct Enzymatic Measurement in Diabetic and Prediabetic Subjects, JOURNAL OF CLINICAL MEDICINE, 12, 1-19,doi:10.3390/jcm12103570
- ***Article - SCI-Expanded***: Ertürk Zararsız G., Bolat S., **Cephe A.**, Kochan N., Yerlitaş S. İ., Doğan H. O., Zararsız G. (2023). Validation of low-density lipoprotein cholesterol equations in pediatric population, PEERJ, 11, 1-17,doi:10.7717/peerj.14544
- ***Article - SCI-Expanded***: **Cephe A.**, Kochan N., Ertürk Zararsız G., Eldem V., Coşgun E., Karabulut E., Zararsız G. (2023). Voomsom: Voom-Based Self-Organizing Maps For Clustering RNA-Sequencing Data, CURRENT BIOINFORMATICS, 18, 1-10,doi:10.2174/1574893618666221205154712
- ***Article - SCI-Expanded***: Ertürk Zararsız G., Bolat S., **Cephe A.**, Koçhan N., Yerlitaş S. İ., Doğan H. O., Zararsız G. (2022). Validation of Friedewald, Martin-Hopkins and Sampson low-density lipoprotein cholesterol equations, PLOS ONE, 17, 1-18,doi:10.1371/journal.pone.0263860
- ***Article - SCI-Expanded***: Özyurt K., Zararsiz G., Ertaş R., **Cephe A.**, Kutlu Ö., Elmas Ö. F., Akkuş M. R., Kutlu F. N., Atasoy M. (2022). Survival of biological therapeutics in psoriasis: Retrospective analysis of 3-years data in a Turkish Registry, PSORTAKSIS, Turkish Journal of Medical Sciences, 52, 58-66,doi:10.3906/sag-2104-339
- ***Article - ESCI***: Ergin M., Özkan M., Ilgaz Ergin M., **Cephe A.** (2022). The Effect of Mutations Determined by Liquid Biopsy on the Progress of the Disease in Advanced Lung Adenocancer, ERCIYES MEDICAL JOURNAL, 44(6), 1-24,doi:10.14744/etd.2022.09514
- ***Article***: Demirtürk B., Bozkuş N., **Cephe A.**, Aktaş B., Taşdemir Afşar S., Çınar H. (2011). Yoksulluk Düzeyinin Modellenmesi Üzerine Bir Araştırma, Journal of Sociological Research, 14(1), 30-67,doi:10.14744/etd.2022.09514
- ***Book Chapter***: **Cephe A.**, Koçhan N., Ertürk Zararsız G., Eldem V., Zararsız G. (2022). Class Discovery, Comparison, and Prediction Methods for RNA-Seq Data, Encyclopedia of Data Science and Machine Learning (3 Volumes), Editor: John Wang, Editor, IGI Global, Pennsylvania, 123-134, doi:10.4018/978-1-7998-9220-5.ch123

- ***Book Chapter***: **Cephe A.**, Koçhan N., Aksel E. G., İpekten F., Yerlitaş S. İ., Ertürk Zararsız G., Zararsız G. (2023). Bioinformatics and Biostatistics in Precision Medicine, Oncology: Genomics, Precision Medicine and Therapeutic Targets, Editors: Hardeep Singh Tuli and Mükerrem Betül Yerer Aycan, Springer, Pennsylvania, ISBN: 978-981-99-1528-6, 189-237, doi: 10.1007/978-981-99-1529-3
- ***Proceedings***: **Cephe A.**, Sezgin A., Koçhan N., Ertürk Zararsız G., Eldem V., Karabulut E., Zararsız G. (2022). RNA-Dizileme Verilerinde Sağlalım Algoritmalarının Kapsamlı Bir Karşılaştırması, 23rd National and 6th International Biostatistics Congress, Ankara, Türkiye, 26 - 29 Oct 2022, p.30
- ***Proceedings***: Ergin M., Özkan M., Ilgaz Ergin M., **Cephe A.** (2022). Metastatik küçük hücreli dışı akciğer kanseri hastalarında likit biyopsi ile çalışılan mutasyon panelinin terapötik ve prognostik önemi, 5th Congress of Internal Medicine, Antalya, Türkiye, 2 - 05 Jun 2022, p.19-22
- ***Proceedings***: **Cephe A.**, Göksülük D., Korkmaz S., Zararsız G. (2022). 2x2 Çapraz Tablolarda İstatistiksel Analizler İçin Kapsamlı Bir R Shiny Uygulaması, Why R? Turkey 2022, Ankara, Türkiye, 15 - 17 Apr 2022, p.20
- ***Proceedings***: **Cephe A.**, Koçhan N., Ertürk Zararsız G., Eldem V., Karabulut E., Zararsız G. (2021). A Novel Approach to Clustering RNA-Sequencing Data Based on Self-Organizing Maps Algorithm, 22nd National and 5th International Biostatistics Congress, Ankara, Türkiye, 28 - 30 Oct 2021, p.51-52
- ***Proceedings***: Zararsız G., Yerlitaş S. İ., İpekten F., Çelik E., **Cephe A.**, Koçhan N., Ertürk Zararsız G., Eldem V., (2021). Eksozom Türevli Non-coding RNA'ların Karakterizasyonunda Biyoinformatik Yaklaşımlar, 22nd International Congress of Parasitology, Aydın, Türkiye, 11 - 15 Oct 2021, p.63-65
- ***Proceedings***: Arıkan T. B., Hamurcu M., **Cephe A.**, Sözüer E. M., Öz A. B., Akyıldız H. Y. (2021). AKUT PANKREATİTLİ HASTALARIMIZIN DEĞERLENDİRİLMESİ, 15th Turkish HPB Surgery Congress and 6th HPB Surgical Nursing Congress, İstanbul, Türkiye, 9 - 12 Sep 2021, p.165
- ***Proceedings***: Zararsız G., Eldem V., Göksülük D., Klaus B., Selçuk K., Ertürk Zararsız G., **Cephe A.**, Öztürk A. (2020). Novel methods and tools for predictive modeling of RNA-sequencing data, The 13th International Symposium on Health Informatics and Bioinformatics, İstanbul, Türkiye, 22 - 23 Oct 2020, p.63
- ***Proceedings***: **Cephe A.**, Zararsız G.(2020). New methods for clustering RNA-sequencing data, The 13th International Symposium on Health Informatics and Bioinformatics, İstanbul, Türkiye, 22 - 23 Oct 2020, p.68
- ***Proceedings***: Ünlüsavuran M., Sönmez C., **Cephe A.**, Eldem V., Ertürk Zararsız G., İpekten F., Zararsız G. (2020). Performance evaluation of automated machine-learning algorithms in omics data, The 13th International Symposium on Health Informatics and Bioinformatics, İstanbul, Türkiye, 22 - 23 Oct 2020, p.84
- ***Proceedings***: **Cephe A.**, Zararsız G., Ertürk Zararsız G., Korkmaz S., Göksülük D., Eldem V., Öztürk A. (2018). RNA-Dizileme Verilerinin Kümelenmesinde Yeni İstatistiksel Yaklaşımlar, XX. NATIONAL and III. INTERNATIONAL CONGRESS OF BIOSTATISTICS, Gaziantep, Türkiye, 26 - 29 Oct 2018, p.14-15
- ***Proceedings***: **Cephe A.**, Zararsız G., Korkmaz S., Bilgin H., Göksülük D., Ertürk Zararsız G., Elmalı F., Öztürk A. (2017). 2x2 Tablolar için Kapsamlı Bir Web Yazılımı, XIX. NATIONAL and II. INTERNATIONAL CONGRESS OF BIOSTATISTICS, Antalya, Türkiye, 25 - 28 Oct 2017, p.106
- ***Proceedings***: **Cephe A.**, Zararsız G., Göksülük D., Korkmaz S., Öztürk A. (2016). DataOrganizer: Veri Manipülasyonu Web Yazılımı, XVIII. NATIONAL AND I. INTERNATIONAL BIOSTATISTICS CONGRESS, Antalya, Türkiye, 26 - 29 Oct 2016, p.58

Projects

- **Erciyes University Research Project - Thesis Project**: Assoc.Prof.Dr. Gökmen Zararsız, New Statistical Approaches in Clustering RNA-Sequencing Data, 2018, This project is supported by Erciyes University BAPSIS.

Honors and Awards

- October 2017, Award for the second best short oral presentation, 19th National and 2nd International Biostatistics Congress of the Biostatistics Association

Volunteer Experience

- **Database Management and Queries with SQL / Using SQL Data in Package Programs** Kayseri, Turkey
  Biostatistics Association Informatics Summer School / https://yazokulu.turcosa.com.tr/ *Jan 2019*
- **Artificial Intelligence Education in Applied Medicine** Kayseri, Turkey
  Cluster Analysis Applications *May 2022*

Skills Summary

- **Languages**: R, Python, PHP, C++, JavaScript, SQL, PL/SQL, JAVA
- **Frameworks**: Scikit, NLTK, SpaCy, TensorFlow, Keras, Django, Flask, NodeJS, LAMP
- **Tools**: Docker, GIT, PostgreSQL, MySQL, SQLite
- **Platforms**: Linux, Web, Windows, IBM Cloud
- **Statistical Programs**: IBM SPSS, SAS, Stata, Minitab, NCSS, Statistica, TURCOSA