TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

Developing Methods for Predictive Data Science

With Applications to Bayesian Inference, Cross-Validation, and Kernel Methods

Riikka Numminen

TURUN YLIOPISTON JULKAISUJA – ANNALES UNIVERSITATIS TURKUENSIS
SARJA – SER. F OSA – TOM. 90 | TECHNICA – INFORMATICA | TURKU 2026

TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

DEVELOPING METHODS FOR PREDICTIVE DATA SCIENCE

With Applications to Bayesian Inference, Cross-Validation, and Kernel Methods

Riikka Numminen

TURUN YLIOPISTON JULKAISUJA ANNALES UNIVERSITATIS TURKUENSIS
SARJA SER. F OSA TOM. 90 | TECHNICA INFORMATICA | TURKU 2026

University of Turku

Faculty of Technology
Department of Computing
Computer Science
Doctoral Programme in Technology (DPT)

Supervised by

Research Director
Professor Jukka Heikkonen
Department of Computing
University of Turku

Associate Professor Antti Airola
Department of Computing
University of Turku

Professor Tapio Pahikkala
Department of Computing
University of Turku

Reviewed by

Professor Hanna Suominen
The Australian National University

Associate Professor Ville Hautamäki
University of Eastern Finland

Opponent

Professor Michael Cochez
Åbo Akademi University

The originality of this publication has been checked in accordance with the University of Turku quality assurance system using the Turnitin OriginalityCheck service.

Language technology tools utilizing artificial intelligence were used to proofread and improve the language of this thesis. The usage of these tools follows the University of Turku guidelines.

ISBN 978-952-02-0775-5 (PRINT)
ISBN 978-952-02-0776-2 (PDF)
ISSN 2736-9390 (PRINT)
ISSN 2736-9684 (ONLINE)
Painosalama, Turku, Finland, 2026

To my family

UNIVERSITY OF TURKU
Faculty of Technology
Department of Computing
Computer Science
NUMMINEN, RIIKKA: Developing Methods for Predictive Data Science
Doctoral dissertation, 116 pp.
Doctoral Programme in Technology (DPT)
September 2026

ABSTRACT

Utilizing large amounts of data requires high-quality and efficient data analytics methods. However, the methods are not universal nor ready-to-use in every case, but instead, existing methods often need to be modified or developed to be suitable for the case. The quality of methods can be assessed by several criteria, and the practical utility of the methods is determined by the amount of resources needed for using them. Method development is often balancing between different characteristics, i.e. some need to be weakened in order to achieve an improvement in another.

This thesis consists of three publications, in which methods of predictive data science were developed. The first method was developed for predicting the profitability of free-to-play mobile games using data collected within a short time interval, and is based on Bayesian inference, survival analysis, and a mixture model. The second method is a quicksort-based acceleration of the tournament leave-pair-out cross-validation for estimating binary classification performance via Receiver Operating Characteristics (ROC) curve analysis. The third method is a learning algorithm that was developed based on the learning algorithm referred to as CGKronRLS by using nonsmooth multiobjective optimization, and its goal is to learn sparse (easier to interpret) hypotheses on pairwise data.

The results obtained with simulated and available real-world data sets were consistent in every research study. The profitability prediction model was observed to be able to predict the monetization percentage, but it was noticed that badly selected prior distribution might lead to biased estimates. The cross-validation method produced equally good estimates of the ROC curves and the areas under them as the earlier, slower method, but it was noticed that the ranking of the data may not be unequivocal with the presented method. The learning algorithm was noticed to learn sparse hypotheses when the decrease in the performance measure was restricted to five percents in comparison to the prediction performance of the reference method, and even sparser when the decrease in prediction performance was not limited. All the methods presented in this thesis improve the existing methods or serve as an alternative for solving these tasks.

KEYWORDS: Bayesian inference, cross-validation, kernel methods, optimization

TURUN YLIOPISTO
Teknillinen tiedekunta
Tietotekniikan laitos
Tietojenkäsittelytiede
NUMMINEN, RIIKKA: Developing Methods for Predictive Data Science
Väitöskirja, 116 s.
Teknologian tohtoriohjelma (DPT)
Syyskuu 2026

TIIVISTELMÄ

Suurten tietomäärien hyödyntäminen vaatii laadukkaita ja tehokkaita data-analytiikan menetelmiä. Ne eivät kuitenkaan ole yleispäteviä, joka tilanteeseen sopivia, vaan usein olemassa olevia menetelmiä joudutaan muokkaamaan tai kehittämään tilanteeseen sopiviksi. Menetelmien laatua voidaan arvioida useilla eri kriteereillä ja niiden vaatimien resurssien määrä vaikuttaa niiden käytännöllisyyteen. Menetelmäkehitys on usein tasapainoilua eri ominaisuuksien välillä, eli jotakin ominaisuutta joudutaan heikentämään, jotta saavutetaan parannus toisen ominaisuuden suhteen.

Tämä väitöskirja koostuu kolmesta osajulkaisusta, joissa kehitettiin ennustavan datatieteen menetelmiä. Ensimmäinen menetelmä kehitettiin ennustamaan ilmaisten mobiilipelien tuottavuutta lyhyellä seuranta-ajalla kerätystä datasta ja se pohjautuu Bayes-päättelyyn, elinaika-analyysiin sekä sekoitemalliin. Toinen menetelmä on pikalajitteluun perustuva nopeutus turnajais-pari-pois-ristiinvalidoinnista, jota voidaan käyttää binääriluokittelijan ennustuskykyä kuvaavan erottelukykykäyrän estimointiin. Kolmantena menetelmänä on CGKronRLS-nimisen oppimisalgoritmin pohjalta, epäsileää monitavoiteoptimointia käyttämällä kehitetty oppimisalgoritmi, jonka tavoitteena on oppia harvoja (helpommin ymmärrettäviä) hypoteeseja parittaisella datalla.

Kussakin osatutkimuksessa saatiin yhteneviä tuloksia simuloiduilla ja valmiiksi olemassa olleilla todellisilla havaintoaineistoilla. Numeerisilla kokeilla osoitettiin menetelmien toimivuus, mutta havaittiin myös jatkokehitystä vaativia asioita. Tuottavuutta ennustavan mallin havaittiin pystyvän arvioimaan todellista maksavien käyttäjien osuutta, mutta huomattiin myös huonosti valitun priorijakauman vaikutus estimaattien harhaan. Ristiinvalidointimenetelmä tuotti yhtä tarkkoja arvioita erottelukykykäyrästä ja sen alle jäävästä pinta-alasta kuin alkuperäinen, hitaampi menetelmä, mutta huomattiin, että tällä menetelmällä saatava havaintojen järjestys ei välttämättä ole yksikäsitteinen. Oppimisalgoritmin havaittiin tuottavan harvoja hypoteeseja silloin, kun ennustuskyvyn heikkeneminen oli rajoitettu viiteen prosenttiin tiheän verrokkimallin ennustuskyvystä, sekä vielä harvempia silloin, kun ennustuskyvyn heikkenemistä ei ollut rajoitettu. Jokainen tässä väitöskirjassa kehitetty menetelmä parantaa olemassa olevia ratkaisuvaihtoehtoja tai tarjoaa uusia näkökulmia näiden ongelmien ratkaisuun.

ASIASANAT: Bayes-päättely, optimointi, ristiinvalidointi, ydinfunktiomenetelmät

Acknowledgements

I wish to express my sincere gratitude to all those who have supported and influenced me during the work leading to this dissertation. The past years have included both successes and challenges, and I am deeply grateful to the many people whose support has made this work possible.

First and foremost, I thank my supervisors, Professor Tapio Pahikkala and Associate Professor Antti Airola, for their expert guidance, encouragement, and the research opportunities they provided throughout this project. I am particularly grateful for the attention you gave to my work; your comments and suggestions significantly improved the thesis. I also thank Professor Hanna Suominen and Associate Professor Ville Hautamäki for serving as preliminary examiners and for their valuable feedback, and Professor Michael Cochez for accepting the role of opponent.

I have also had the privilege of working with excellent co-authors. I thank Markus Viljanen for being my first point of contact with this research group and for encouraging me to pursue doctoral studies here. I thank Ileana Montoya Perez and Napsu Karmitsa for fruitful collaboration and insightful discussions, which helped me better understand academic research and its practices. I also thank Pauliina Paasivirta for her work on the original version of Publication III, which provided an important foundation for my subsequent work after she moved from academia to other professional responsibilities.

I am grateful to many colleagues for the supportive and stimulating environment in which this research was carried out. Conversations in the office, in the coffee room, and over 'Friday lunches' have offered perspective and encouragement at various stages of the process. In particular, I thank Parisa Movahedi for peer support during periods when I struggled with thesis writing and with balancing different aspects of life. I also thank Katariina Perkonoja and Luca Zelioli for discussions that helped me clarify my thinking as the thesis developed, and Katri Karhinoja for steadfast support and for being there when it mattered. The Department of Computing floorball sessions have been an important counterbalance to academic work and a valuable way to manage stress; I therefore thank Erno Lokkila for organizing them and everyone I had the opportunity to play with. While I cannot mention everyone individually, I am sincerely grateful to all those who contributed to the everyday sense of community through coffee breaks, 'Friday lunches', and floorball.

I gratefully acknowledge the funding provided by the Doctoral Programme in Mathematics and Computer Sciences through a funded doctoral researcher position, as well as financial support from the Nokia Foundation in the form of a Nokia Scholarship.

Finally, I wish to thank my son Leo, my partner Marko, and my family and friends. Your support, in all its forms, has sustained me throughout this journey.

17.6.2026
Riikka Numminen

Table of Contents

Acronyms

AIC	Akaike Information Criterion
AUC	Area under the ROC curve
C-index	Concordance index
EM	Expectation Maximization
ERM	Empirical Risk Minimization
FPR	False Positive Rate
LMBM	Limited Memory Bundle Method
LOO	Leave-one-out (cross-validation)
LPO	Leave-pair-out (cross-validation)
MAP	Maximum a Posteriori
MLE	Maximum Likelihood Estimation
QLPO	Quicksort leave-pair-out (cross-validation)
RKHS	Reproducing Kernel Hilbert Space
RLS	Regularized least-squares
ROC	Receiver Operating Characteristics
SRM	Structural Risk Minimization
SVM	Support vector machine
TLPO	Tournament leave-pair-out (cross-validation)
TPR	True Positive Rate

Nomenclature

$\mathbf{a}$	Vector of dual coefficients
C	Random variable denoting censoring time
$\mathcal{D}$	Unknown data distribution
$\mathbb{E}$	Expectation
f	True unknown function
$\hat{f}$	Estimate of the true unknown function
$\mathcal{F}$	Feature space
h	A hypothesis in the hypothesis space
$\mathcal{H}$	Hypothesis space
$\mathcal{I}$	Set of indices of a sample, also used for referring to the sample
$\mathcal{I}_{test}, \mathcal{I}_+, \mathcal{I}_-$	Subsets of the sample
$\mathbb{I}$	Indicator function
K	Kernel function
L	Likelihood function
l	Log-likelihood function
$\mathcal{L}$	Error function, empirical risk
ℓ	Loss function
$\ell_0, \ell_1, \ell_2, \ell_{[k]}$	Norms, also marked by $\|\|\cdot\|\|$
n	Sample size, $\|\mathcal{I}\|$
p	Density function of a probability distribution
$\mathbb{P}$	Probability function
R	Complexity/regularization function
S	Survival function
X	Random variable or vector of independent variable(s)
x	Realization of X
$\mathcal{X}$	Set of possible values X can have
Y	Random variable or vector of dependent variable(s)
y	Realization of Y
$\mathcal{Y}$	Set of possible values Y can have
Z	Random variable or vector of all variables
z	Realization of Z
$\boldsymbol{z}$	Data sample
$\mathcal{Z}$	Set of possible values Z can have

Z^{obs}	Observed incomplete data
α	Scaling coefficient between two parts of a model
γ	Threshold value
δ	Random variable and its realization denoting censoring indicator
ε	Irreducible error
Θ	Asymptotically bounded time complexity function
θ	Model parameter(s) to be optimized
$\hat{\theta}$	Optimal value(s) for the model parameter(s)
κ_γ	Classifier
ρ	Regularization coefficient
T	Random variable denoting time
τ	Realization of T
T^*	Random variable denoting event time
Ω	Set of possible values for the model parameter(s)

List of Original Publications

This dissertation is based on the following original publications, which are referred to in the text by their Roman numerals:

I Riikka Numminen, Markus Viljanen and Tapio Pahikkala. Bayesian inference for predicting the monetization percentage in free-to-play games. IEEE Transactions on Games, 2022; 14(1): 13–22.

II Riikka Numminen, Ileana Montoya Perez, Ivan Jambor, Tapio Pahikkala and Antti Airola. Quicksort leave-pair-out cross-validation for ROC curve analysis. Computational Statistics, 2023; 38: 1579–1595.

III Pauliina Paasivirta, Riikka Numminen, Antti Airola, Napsu Karmitsa and Tapio Pahikkala. Predicting pairwise interaction affinities with ℓ_0-penalized least-squares—a nonsmooth bi-objective optimization based approach. Optimization Methods and Software, 2026; 41(2): 450–477.

1 Introduction

Data science is using data to answer scientific questions. It has been defined as a discipline that uses methods from statistics, informatics, computing, communication, management, and sociology to examine data and turn it into useful knowledge and insights [1]. As is visually explained in Figure 1, it is a field that is at the centre of the intersection of mathematics and statistics, computer science, and a field of application. Data science can be viewed from ethical, computational, statistical, scientific and human perspectives [2]. Ethical perspective is the foundation providing guidelines for good practice. Scientific, statistical and computational perspectives correspond to formulating a problem, designing a study and analysis, and providing the used algorithms, respectively. Human perspective brings everything together by providing the aspects of communication during the work between the people involved and afterwards communicating the results to the other people [2]. These perspectives are visualized in Figure 2 in a way that the scientific, statistical and computational perspectives form the core of data science and are surrounded by the ethical and human perspectives, which influence all parts of it.

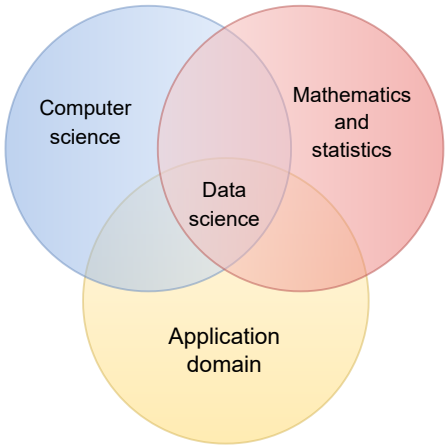

Figure 1. Data science in relation to other fields.

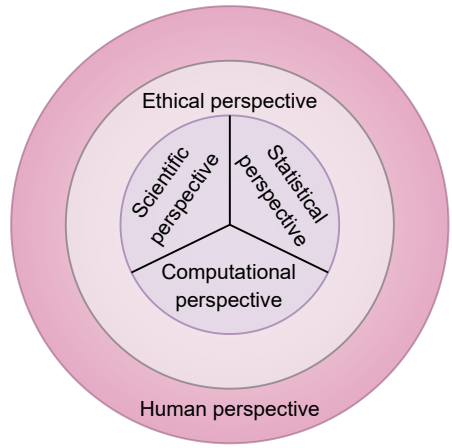

Figure 2. Different perspectives [2] to view data science.

When it comes to methods, the focus is on statistics and computer science, as data science can be viewed as a child of them [3]. A sample of data is observed from

a field of application and then selected methods are used for e.g. interpreting and explaining the data sample, inferring from the sample to population level, or making predictions for new data based on the sample. Different aims of data science can be split into three categories [1] based on a question that is being answered to: descriptive analytics focus on the past and present by explaining what has happened or is happening, whereas predictive and prescriptive data analysis focus on the future by answering questions 'what will happen?' and 'how to make it happen?'. Which methods are selected depends on the question that is being answered. Descriptive analytics are conducted by e.g. visualization, summarizing the data by central moments and using correlation analysis to discover connections between variables in data [4]. In predictive analytics, methods like machine learning, statistical learning, statistical inference and survival analysis are used to predict a more general case based on the observed sample [5]. Prescriptive analytics, then again, go even further by recommending actions to be taken based on the outcomes of the predictive models. This can be done by e.g. simulation, decision analysis [6], stochastic modelling [7] or reinforcement learning [8]. The focus of this thesis is on methods for predictive analytics. More precisely, the methods used and developed in the studies are either in the field of machine learning or statistical inference.

Machine learning methods and methods for statistical inference are partly overlapping, but the observed sample is used for different purposes in those fields. They correspond to the two cultures of statistical modelling presented by Breiman [9]: statistical inference to the data modelling culture and machine learning to the algorithmic modelling culture. In statistical inference, the goal is to describe and understand the population better by estimating the population distribution based on the observed sample. Then again, in machine learning, the goal is to estimate the function that represents the true connection between the input and output variables, so that the estimate can be used for predicting the labels of new data points. These points-of-view have also been called generative and discriminative approaches of machine learning [10], and in both of them, the main components are the same, but they might represent different things.

Method development can be done in different parts of the process and related to the different perspectives of data science. It can be in the scientific perspective by formulating a problem in a field of application so that methods included in the statistical and computational perspectives that are used in other fields can be modified to be suitable for use in this new field. It can be in the statistical or computational perspective, meaning modifying or improving a model's formulation or the algorithm used for finding the final hypothesis, or the performance evaluation measures. Rarely is anything completely new developed, as usually all the methods are based on something already existing.

When talking about improving a method or coming up with a new method for solving a specific task, it is important to have clear criteria for evaluating the quality

of the method. Aliferis and Simon [11] have formalized different quality aspects of a data science method, which are also possible areas for improvement. The quality aspects are representation power, transparency, soundness, completeness, computational complexity for learning or using a model, other cost functions, space complexity for learning, storing or using a model, sample complexity, learning curves and power-sample requirements, and probability and decision theory consistency [11]. These quality aspects are versatile and rigorously define what to consider while evaluating a model or improvement.

The importance of method development on a general level is self-evident. First of all, new problems can be solved. Secondly, the better and more reliable the data science methods are, the better they can be utilized and taken advantage of in society and the everyday lives of the people. These aspects are measured by soundness and completeness. In addition, transparency and explainability are other important aspects of methods and are highly related to the people trusting them – especially if they are aimed to be used in sensitive fields where it is important to understand the reasoning behind the decisions given by a model. This aspect is a great weakness of for example deep learning methods, which are used increasingly. The faster to compute and more memory-efficient the methods are, the fewer resources are needed for using them. The computational aspects are also related to financial resources as well as even to their carbon footprint and sustainability of using them. Sample complexity, learning curve and power-sample requirements of a method are related to the amount of data that is needed. Even while living in the era of big data, collecting data can be time-consuming and require a lot of financial resources in some fields. Thus, the smaller the sample complexity of a method is, the more practical it is and the wider it can be applied in different fields.

This thesis provides insight into data science method development through several studies. A common goal in the studies was to present alternative methods for solving some prediction problems in the fields of application. Nevertheless, the field of application, the type of method development and the method development aspect varied from study to study, and the motivation for the developed methods came from the fields of application. The types of method development in the studies included in this thesis are:

- Defining a problem of estimating the profitability of free-to-play mobile games within a survival analysis framework, and utilizing Bayesian inference to predict the values of the parameters of the model. The model enables a preliminary estimation of the profitability without having extensive historical data available.

- Introducing a variation of a cross-validation method, whose time complexity for obtaining a full ranking of data needed for ROC curve analysis is lower than that of a previously existing method for the same task. The presented

method reduces, on average, the number of cross-validation rounds and thus the number of learned hypotheses, which makes it particularly suitable for scenarios where model training is computationally intensive.

- Presenting a well-defined learning problem and an algorithm for solving it, in order to learn sparse hypotheses on pairwise data without compromising the prediction performance considerably. Having sparse hypotheses enhances interpretability and improves time efficiency in prediction because only the most important training data observations are considered in the model.

These topics were in part determined by the projects during which the studies were conducted. The mobile game analytics research was conducted on data provided by a Finnish free-to-play mobile game developer, and the other two studies were conducted as part of a Research Council of Finland project, where the aim was to develop tools to support machine learning and model evaluation in precision medicine applications.

This thesis consists of two parts. The first part is formed by Chapters 1-5 as follows: It is started by an introduction to the thesis in Chapter 1, which is followed by presenting the theoretical foundation of predictive data science in Chapter 2. In the theory part, the scope is limited to the concepts that are needed for understanding the studies. After that, the quality aspects and a process of method development are described in more detail in Chapter 3. In Chapter 4, the research studies are presented according to the method development process. Finally, the first part of the thesis is concluded, and future research is discussed in Chapter 5. After that begins the second part, where the original publications included in this thesis are gathered.

2 Predictive Data Science

Predictive data science is a field where a sample of data is observed and used to select an optimal hypothesis from a set of possible hypotheses, and the hypothesis is then used to make predictions. Generative and discriminative approaches [10] are two viewpoints of data science with different goals. In the generative approach, the observed data are assumed to follow a probability distribution, and the goal is to estimate the density function related to it. Parametric density estimation is a parametric method for achieving this goal, whereas kernel density estimation and generative neural networks are nonparametric methods. On the other hand, in the discriminative approach, there are no assumptions about the underlying distribution of the data; instead, the goal is to learn a hypothesis that accurately predicts on data that was not seen during training [10].

The predictive data science process consists of several components, which are described as a flowchart in Figure 3 for both data science approaches. The starting point is to have a sample of data. Then, a learning procedure is conducted to obtain candidate hypotheses, among which a final hypothesis is selected by model selection. Depending on the data science approach, the model selection is conducted either on the same data from which the hypotheses were inferred or on a separate validation dataset. Eventually, the generalization performance of the selected final hypothesis is evaluated on a separate test dataset in the discriminative approach.

As only estimates, instead of the true values, of the underlying unknown reality and performance can be obtained, it is important to understand the characteristics of the estimates – especially their reliability and stability. A good estimator produces values that are near the corresponding true value: the closer they concentrate around the true value, the better the estimator is. An estimate is unbiased when its expected value is equal to the true value, i.e. there is no systematic shift in how the estimates calculated from different data samples are scattered around the true value. Furthermore, efficiency is another important characteristic, as it measures the range of the area where the estimates are [12].

Practical utility is another equally important characteristic of an estimator, in addition to its unbiasedness and variance [12]. Practical utility can be measured by the need for different resources, e.g. computational (time), space (memory) and sample complexities [11]. Both the learning and evaluation phases of the data science process are computational procedures that transform an input into an output, and

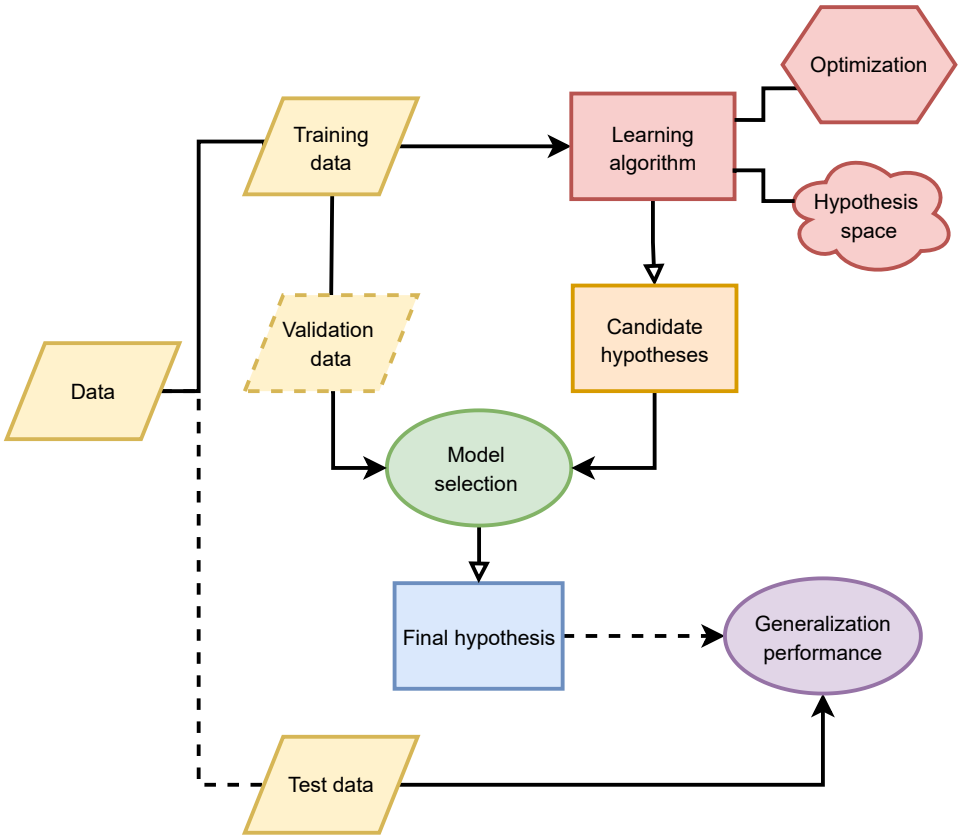

Figure 3. Components of a predictive data science process. Dashed lines mean that a separate test set is taken, and the generalization performance of a final hypothesis is evaluated only in the discriminative approach. A dashed rhombus means that separate validation data is included in the process only in the discriminative approach.

thus, they fulfil the informal definition of an algorithm [13], and their efficiency can be measured similarly to the efficiency of algorithms. The complexities are often measured as functions of the input size. Asymptotic order of growth of any resource requirements is denoted by a function Θ [13].

The methods developed in the research studies followed different data science approaches, were associated with different components of the flowchart in Figure 3, and utilised different existing theories and methods. A publication-wise summary of these is given in Table 1, and more information about the relevant topics is given in the following sections: More information about data and the data science approaches on a general level is given in Section 2.1. That is followed by explaining the components related to the learning phase in Section 2.2. Both model selection and evaluation of generalization performance are discussed in Section 2.3 along with resampling methods.

Table 1. Summary of the data science approaches, flowchart components in Figure 3 to which the method development is related, and theory topics that are used in the publications.

Publication	Data science approach	Flowchart component	Relevant theory
I	Generative	Hypothesis space	Time-to-event data, survival analysis, MLE, MAP, EM-algorithm
II	Discriminative	Model selection, generalization performance	Cross-validation, ROC curve, AUC, quicksort, computational complexity, estimation theory
III	Discriminative	Learning algorithm – optimization	Pairwise data, SRM, LMBM, sparsity, C-index

2.1 Data

In a data science task, a sample of data is observed and assumed to follow a probability distribution $\mathcal{D}$. The sample provides partial information about $\mathcal{D}$ over a population of interest to the learner [10]. The population is described by some measurable characteristics, which are called random variables and denoted in this thesis by Z. The set of possible values for Z is $\mathcal{Z}$, and its content depends on the type of the variables. They can be quantitative or qualitative, but in predictive data science, the qualitative variables are usually represented in a numeric form. The quantitative vari-

ables can be continuous or discrete, which can be further categorized as categorical, ordinal, or nominal depending on whether the discrete values have a natural ordering or not [4]. For continuous random variables, $\mathcal{Z}$ is usually assumed to consist of real numbers $\mathbb{R}$, and for discrete variables, it is typically a subset of natural numbers $\mathbb{N}$. In this thesis, the sample of observations is denoted either by $\boldsymbol{z}$ (a sequence of realizations of Z), when the observed values are emphasised, or with an index set notation $\mathcal{I} = \{1, \dots n\}$, when the focus is on which data points are included. In other words, a datum $\boldsymbol{z}_i$ can be referred to just by the index $i \in \mathcal{I}$.

The random variable follows a probability distribution $\mathcal{D}$, whose probability density function is denoted as p_Z or without the subscript if it is clear from the context. In addition, p_Z denotes a joint probability distribution when Z is a multivariate random variable. It is assumed in data science that the sample $\mathcal{I}$ is independent and identically distributed (i.i.d.) according to $\mathcal{D}$. In other words, the examples in the sample are independent of each other, and they are sampled from the same distribution. In the generative approach, the goal is to estimate the parameters of $\mathcal{D}$, i.e. to understand the distribution from which the data are sampled. Depending on the context, the notation $\mathcal{D}$ or p is used to refer to the distribution. Furthermore, they are connected to a general notation of probability $\mathbb{P}$:

$$\mathcal{D}(A) = \mathbb{P}_{z \sim \mathcal{D}}[Z \in A] = \int_A p_Z(z) dz$$

for an event $A \subset \mathcal{Z}$, expressing how likely it is to have an observation $z \in A$ [10].

In this thesis, the data used in the studies followed mainly a supervised learning paradigm, where it is usually written that $Z = (X, Y)$, because the data consists of both independent and dependent variables. Notations X and Y, respectively, are used to refer to them, and the realizations of them are x and y. They are also called input and output variables, or features and labels, again in the corresponding order. Similarly to the previous paragraph, the sets of possible values for X and Y are $\mathcal{X}$ and $\mathcal{Y}$, respectively. If the labels are known only for some of the examples in the sample, the paradigm is called semisupervised, and there are additional challenges as a consequence, which is seen in the case of Publication I.

In supervised learning using the discriminative approach, the goal is to estimate a function $f \colon \mathcal{X} \to \mathcal{Y}$, which describes the unknown underlying connection between the input and output variables, and eventually use the estimate for predicting the labels of new data. More precisely, it can be assumed that the underlying reality can be modelled as $Y = f(X) + \varepsilon$, where ε is an error term that contains everything that is not modelled by f, also called an irreducible error [5]. These kinds of learning tasks are usually regression or classification problems, depending on the content of $\mathcal{Y}$. In a regression task, the labels Y are continuous, e.g. $\mathcal{Y} = \mathbb{R}$. In contrast, it is a classification task when $\mathcal{Y}$ consists of predefined categories.

Depending on the application domain, the data may exhibit special characteristics arising from the nature of the features, the data-collection process, or dependencies between variables. In Publications I and III, for example, the data were time-to-event and pairwise, respectively. These characteristics are briefly discussed in the following Sections 2.1.1 and 2.1.2.

2.1.1 Time-to-Event Data and Survival Analysis

Time-to-event data measures the time it takes for an event of interest to occur for the individuals in the sample. The type of the event depends on the field of application. This kind of data appears, for example, in medicine, where the event would be e.g. a diagnosis of a disease or the death of a patient, and in industry, where the event would be e.g. some part of a machine to break. The time when the event happens is called an event time, and the time until that, the follow-up time. As the follow-up happens within a limited time interval, it is not always possible to know the exact times of entering the study or having the event, but instead it is only known that the entry has happened before the follow-up started or the event was to happen later than when the follow-up ended. In these cases, the data are left- or right-censored, respectively [14].

Another example of a field of application where this kind of data exists is game analytics, where the interest is often in player retention, churn rate and monetization (see the review [15] and the reference therein for more information). In particular, when it comes to estimating the profitability of a game, as in Publication I, it is of great importance to be able to obtain the estimates as early as possible without having to collect data from a long time interval in order to avoid wasting resources on something that is not even expected to be profitable. Then the event of interest is an event of monetization, and the data are right-censored as the estimation is done before having the historical data.

Survival analysis is a field of statistics that is used for predicting when an individual will experience the event. In survival analysis, it is assumed that every individual will eventually have the event [14], which, however, is not always the case and then a mixture cure model [16] is used instead. In this case, the population is a mixture of two subpopulations: some will have the event at some point, but may not have had it yet, and some will not have the event, no matter how long they were followed, i.e. they are cured. The existence of the two subpopulations affects how to handle censoring, because often it is not possible to know whether the censoring occurred because an individual was not followed long enough or the individual is cured.

In the case of a mixture cure model, a triplet of random variables $Z = (\mathrm{T}, \delta, Y)$ is used to describe the situation. The first two are used in survival analysis, and the third is a label denoting which subpopulation the observation belongs to. The first variable $\mathrm{T} = \min(\mathrm{T}^*, C)$ is the observed time that is either the event time T^* or

the censoring time C, which ever is smaller. The second variable $\delta = \mathbb{I}(C < \mathrm{T}^*)$ is an indicator denoting whether the event was observed during the follow-up $(\delta = 0)$ or the individual was censored $(\delta = 1)$ [14]. The third variable Y is an indicator representing whether the individual is uncured $(Y = 0)$ and will have the event at some point, or if the individual is cured and will be always censored no matter how long they are followed $(Y = 1)$ [16]. Thus, Y is partly latent, because it is known only for those subjects who have had the event before the end of follow-up, i.e. its value is known through the value of δ. This is the definition used by Dempster et al. [17] for the observed data being incomplete. Then again, if the label was known for every subject, the data were said to be complete as the triplet Z would be fully known. In this thesis, the notation Z^{obs} is used to denote the incomplete data, when it is necessary to discriminate between the incomplete and complete data, and otherwise the complete data notation Z is used.

2.1.2 Pairwise Data

Pairwise data is such data, where $\mathcal{X}$ is a Cartesian product of two domain spaces, for example sets of drugs and targets or sets of clients and products. When it comes to this kind of data, the goal is often to predict some joint or relational feature between the domains. To summarise, this means that there are three random variables used for describing the case: one denoting the element in the first domain, another denoting the element in the second domain, and a third being the label representing whether there exists a relation between the two elements (binary label) or the strength of the relation (continuous label). In addition to the identity information that can be derived from the label matrix, there may be some domain-specific side information that helps in making predictions for new data.

A fundamental assumption of pairwise data is that the elements in the domains exist as components of several pairs. Usually, the numbers of unique elements in the domains are much less than the number of pairs. This enables some computational shortcuts that have been discussed for example by Viljanen et al. [18]. Nevertheless, it is possible that not all possible pairs are known, and hence the label matrix can be sparse.

As a consequence of the pairwise assumption, there are four different cases of generalization [19]. The simplest case of generalization is when both elements of test pairs are such that they are included in the training data, but the labels of the exact pairs are not known. Another case is that only one of the elements of the test pairs is seen during training, and the other element is not. The known element can be either of the first or second component of the pair, so there are two different cases in this generalization category. The last case is when neither of the elements of the test pairs is seen during training. This is called a cold-start problem and is the most challenging of the generalization tasks.

2.2 Learning Phase

The main component of the learning phase in Figure 3 is a learning algorithm, and optimization and hypothesis space are highlighted as its core elements. The term learning algorithm is used for the whole learning procedure, also when referring to the generative approach, because the components are the same regardless of the data science approach, although this term is not generally used in statistical literature. In addition, the data science approach has an effect on the contents of these components. Hypothesis space represents what kind of models are aimed to be fitted to the data, which can be considered as a formulation of prior knowledge or assumptions about the data-generating process through p or the mapping f. As the goals are different in the two data science approaches, the commonly solved optimization problems also vary depending on the approach. In the generative approach, the parametric density estimation is often conducted via maximum likelihood estimation or the corresponding Bayesian point-of-view for solving the same task, maximum a posteriori estimation. Then again, when learning a hypothesis in the discriminative approach, the goal of the optimization problem is to minimize an objective function representing the in-sample loss. Furthermore, the used optimization methods depend on the optimization problem. General level characteristics of models are described in Section 2.2.1 together with the two more specific hypothesis spaces that were used in Publications I and III. After that, commonly used optimization problems and algorithms for solving them are described in Sections 2.2.2 and 2.2.3. Finally, examples of learning algorithms used in the Publications are presented in Section 2.2.4.

2.2.1 Hypothesis Space

The learning phase begins by formalising an idea of what kind of model could estimate f or the distribution $\mathcal{D}$ appropriately. This represents prior knowledge or assumptions about the underlying, unknown reality, and the set of possible hypotheses is called a hypothesis space $\mathcal{H}$. Restricting the search space to a certain kind of models, i.e. making assumptions about the structure of the hypothesis space, is done in order to avoid overfitting that could be caused by training data being misleading to the learner if any kind of model is allowed. One type of this kind of prior knowledge is the assumption that the data distribution belongs to a family of parametric distributions. This is used especially in the generative approach. Another type of prior knowledge is an assumption that the hypothesis class fulfils certain learnability criteria for the distribution over the data. The learnability criteria determine the minimum amount of data needed for inferring a good hypothesis, i.e. the sample complexity, in the discriminative approach. However, if the model is not learnable with a certain distribution, the learning process might not converge to a good hypothesis, no matter how much data there is available [10].

A general rule of thumb is to use as simple models as possible. If the complexity of the hypothesis space is high, the models tend to overfit the training data and are not usually very good at generalizing to new data [10]. However, there are also exceptions: Over-parameterized models, e.g. neural networks, have recently been shown to be good at generalization despite interpolating the training data. This occurs as a consequence of a phenomenon called double descent [20]. The more flexible the hypothesis space, the smaller the bias of the model, but the higher the variance when generalizing to new data. Thus, the selection of a hypothesis space is a trade-off between bias and variance or bias and complexity, depending on the point-of-view [10]. In addition to the theoretical properties of the complexity of the model, simpler models are easier to understand, interpret and explain. Explainability is difficult to obtain for complex models, which is a downside of using neural networks, for example. The complexity of the model is strongly dependent on its structure.

The structure of the hypothesis space determines the form of the model. It may be linear or nonlinear, parametric or nonparametric, or even such that it cannot be formalized by an explicit function. Hypothesis space is said to be linear when the hypotheses h in it are linear with respect to the features, e.g. $h(x) = \theta x$, and nonlinear when there is a nonlinear transformation of the features, e.g. $h(x) = \theta x^2$. However, in both of these cases, the models can be linear with respect to the parameters, which is what is called a linear model. Hypothesis space is said to be parametric when there is a fixed number of parameters and the number does not depend on the amount of data available, whereas in a nonparametric case, the number of parameters is not fixed and may depend on the size of the data. Models that cannot be formalized by an explicit function are also nonparametric, because there is no functional form containing parameters whose values could be optimized. Then the structure of the model is based on the used algorithm.

Reproducing Kernel Hilbert Space

Sometimes the input space $\mathcal{X}$ is mapped into another, higher-dimensional space, called a feature space, $\mathcal{F}$, in order to ease the learning task. This is beneficial in that it is then possible to use a linear model in $\mathcal{F}$ while the behaviour can be highly nonlinear in $\mathcal{X}$. However, it comes with a cost: computational complexity may be high for learning linear models over high-dimensional data. Kernel-based learning is used to solve this concern. A kernel function is defined as an inner product in the feature space, and thus, it can be considered as a similarity function. Kernel methods utilize the kernel functions for avoiding the need to explicitly express the mapping from $\mathcal{X}$ to $\mathcal{F}$ [10]. Further, using the kernel trick is not limited to a data of a single table input space only, but instead, it can be applied to structured data as well. Thus, any kind of input space (e.g. graphs, strings, trees) equipped with a positive definite kernel can be mapped into a Hilbert space [21].

Reproducing kernel Hilbert space (RKHS) is a special Hilbert space of functions associated with a positive definite kernel function and has a property that pointwise images of two functions in it are close if the functions themselves are close when the distance measure is based on an inner product [22]. Kernel functions $K(x, x_i)$ are also called representers of evaluation at a fixed value, when they are functions of the first element and the second element is fixed. This is the core of the reproducing property of RKHS: the inner product of two representers of evaluation produces the kernel at the two fixed points. This kernel property provides a finite-dimensional solution to an infinite-dimensional structural risk minimization problem (7) to be presented in Section 2.2.2 [5]:

$$h(x) = \sum_{i=1}^{n} \mathbf{a}_i K(x, x_i), \tag{1}$$

where $\mathbf{a}_i$ are the parameters to be optimized on the dual problem.

Mixture Cure Model

Also the mixture model mentioned in Section 2.1.1 can be viewed as a kind of kernel method [5]. Then the mixture cure model as a function of observed time τ, has the form $h(\tau) = \alpha_0 S_0(\tau) + \alpha_1 S_1(\tau)$, where the mixing proportions α_m (with $m \in \{0, 1\}$) are equivalent to the probabilities of the subpopulations $\mathbb{P}(Y = m)$, and $S_m(\tau) = \mathbb{P}(\mathrm{T}^* > \tau \mid Y = m)$ are called survival functions [14]. The hypothesis is then an unconditional survival function

$$h(\tau) = \mathbb{P}(Y = 0)\,\mathbb{P}(\mathrm{T}^* > \tau \mid Y = 0) + \mathbb{P}(Y = 1)\,\mathbb{P}(\mathrm{T}^* > \tau \mid Y = 1),$$

which reduces to

$$h(\tau) = \pi S_0(\tau) + 1 - \pi, \tag{2}$$

when $\mathbb{P}(Y = 0) = \pi$, S_0 denotes the survival function of the uncured population, and $S_1(\tau) \equiv 1$ [16].

2.2.2 Optimization Problems

The type of the used optimization problem depends on the data science approach. In the generative approach, the goal is to find a probability distribution that fits the data the best, i.e. to find a distribution that is the most likely to have generated the sample. There are two commonly used methods to learn the optimal parameters of the distribution: maximum likelihood estimation and maximum a posteriori estimation. In contrast, in the discriminative approach, the goal is to find a hypothesis that estimates the best the unknown f. This can be done by following empirical or structural

risk minimization learning paradigms. Again, the notation is slightly different depending on the data science approach. In particular, in the generative approach, the optimal parameters are highlighted by using notation $\hat{\theta}$, whereas in the discriminative approach, the final hypothesis is denoted as $\hat{f}$.

Maximum Likelihood Estimation

Maximum likelihood estimation (MLE) is a method used in the parametric density estimation problem for finding such values for the parameters of the probability density function that maximize the value of a likelihood function. Likelihood function is the joint density function of the sample points [4]:

$$L(\theta \mid \boldsymbol{z}) = \prod_{i=1}^{n} p(z_i \mid \theta), \tag{3}$$

where p is a probability density function of the probability distribution from which the data are drawn, parameterized by θ. The probability distribution is not known, but instead, there is usually an idea about the family of distributions that it belongs to. In other words, there is prior knowledge about the model's parametric form, but the values of the parameters need to be inferred from the data sample through the density function. Likelihood function is a function of parameters while having the data fixed, and its value is the likelihood of having observed the given sample of data under the model p. Even though in the likelihood function the data are fixed, in general the data are considered as random variables and the parameters of the density distribution as unknown but fixed. Thus, the goal is to find such values for the parameters that make it the most probable to draw the sample given the parameters.

The model learning is done by finding such $\theta \in \Omega$, where Ω is the set of possible values for the parameters, that maximize the likelihood function. Often it is easier to maximize the logarithm of the likelihood function instead of the likelihood function itself, because taking the logarithm transforms the product in equation (3) into a sum:

$$l(\theta \mid \boldsymbol{z}) = \sum_{i=1}^{n} \log p(z_i \mid \theta).$$

In addition to easing up the differentiation, the transformation from a product to a sum of probabilities makes the computation more feasible with small probabilities due to issues related to the floating point representation (see for example [23] for more information about the floating point representation of numbers). From the monotonicity of the logarithm, it then follows that the optimization problem has the following form:

$$\hat{\theta} = \arg\max_{\theta \in \Omega} L(\theta \mid \boldsymbol{z}) = \arg\max_{\theta \in \Omega} l(\theta \mid \boldsymbol{z}). \tag{4}$$

Maximum a Posteriori Estimation

Maximum a posteriori (MAP) estimation is another method for conducting parametric density estimation. It is used when the Bayesian interpretation of statistics is used instead of the frequentist interpretation, which is related to the MLE. The main differences between Bayesian and frequentist interpretations are that the Bayesians consider probabilities as degrees of belief of events based on the information available, whereas frequentists consider them as frequencies over repeated trials, and the Bayesians view the parameters as random variables, unlike the frequentists, who view them as fixed but unknown values. As the parameter θ is considered as a random variable in the Bayesian inference, there is a probability distribution it follows and a density function $p(\theta)$ characterizing it. This density function is called a prior distribution. Joint probability distribution of the parameters and data $p(\theta, \boldsymbol{z})$ is defined by the prior distribution together with sampling distribution $p(\boldsymbol{z} \mid \theta)$. The joint distribution $p(\theta, \boldsymbol{z}) = p(\theta)p(\boldsymbol{z} \mid \theta)$. Bayes' rule then implies the posteriori distribution for θ:

$$p(\theta \mid \boldsymbol{z}) = \frac{p(\theta, \boldsymbol{z})}{p(\boldsymbol{z})} = \frac{p(\theta)p(\boldsymbol{z} \mid \theta)}{p(\boldsymbol{z})}, \tag{5}$$

where $p(\boldsymbol{z})$ is the marginal distribution of the observed data that can also be called the prior predictive distribution [24].

The goal in MAP estimation is to find such $\theta \in \Omega$ that maximizes the posterior distribution (5), i.e. finds the mode of the distribution. The equation can be simplified for optimization in form $p(\theta \mid \boldsymbol{z}) \propto p(\theta)p(\boldsymbol{z} \mid \theta)$, because the marginal distribution of the data does not depend on θ and hence $p(\boldsymbol{z})$ in (5) does not have an effect in the optimization problem. In addition, the sampling distribution is equal to the likelihood function as it is taken as a function of θ, not $\boldsymbol{z}$, i.e. $p(\boldsymbol{z} \mid \theta) = L(\theta \mid \boldsymbol{z})$ when using the notations presented in the previous section. Then, the optimization problem has the form

$$\hat{\theta} = \arg\max_{\theta \in \Omega} p(\theta)L(\theta \mid \boldsymbol{z}) = \arg\max_{\theta \in \Omega} p(\theta)l(\theta \mid \boldsymbol{z}), \tag{6}$$

where the latter equality follows again from the monotonicity of the logarithm. It can be seen that it is similar to the MLE optimization problem (4), the only difference being that the prior distribution is included. It is also worth noticing that this reduces to (4) if the prior distribution is a uniform distribution [24].

Empirical and Structural Risk Minimization

Empirical risk minimization (ERM) is a learning paradigm used in the discriminative approach when training a supervised machine learning model. In ERM, the goal is to find a hypothesis $h \in \mathcal{H}$ that minimizes a training error $\mathcal{L}_{\mathcal{I}}(h) = \mathbb{E}\left\{\ell(Y, h\left(X\right))\right\}$ [10], where the subscript $\mathcal{I}$ highlights that it is conditioned on the sample of data used for training the model and $\mathbb{E}$ denotes the expected value. Function $\mathcal{L}$ is called

an error function, and it calculates an average of a loss function ℓ over the training data, as the average is used to estimate the expectation.

The optimization problem is then

$$\hat{f} = \arg\min_{h \in \mathcal{H}} \mathcal{L}_{\mathcal{I}}(h),$$

where $\mathcal{H}$ is a set of predictors chosen in advance before seeing the data. Inductive bias is introduced when restricting the set from which the learner can choose the hypothesis. Ideally, the restriction is based on prior knowledge about the problem to be learned, and it is done in order to avoid overfitting. Realizability assumption implies that there exists such a hypothesis h^* whose empirical risk $\mathcal{L}_{\mathcal{I}}(h^*) = 0$, when the examples in $\mathcal{I}$ are sampled according to $\mathcal{D}$ and labelled by f. However, the aim is to measure the true risk $\mathcal{L}_{(\mathcal{D},f)}(h_{\mathcal{I}})$, which is a random variable because there is randomness in the sampling process and hence also in the choice of the predictor $h_{\mathcal{I}}$ [10].

Structural risk minimization (SRM) is a learning paradigm similar to ERM, but the goal of the search is to find a hypothesis that balances between the empirical risk and model complexity. Then the aim is to have a hypothesis whose generalizability is good. In the ERM paradigm, there is a risk of overfitting the training data, which would mean that the empirical risk may be small, but the model fails to generalize to new data. Overfitting can be avoided by an additional regularization term in the objective function. Regularized ERM is an example of SRM, as the regularization restricts the model's complexity. This is also called a regularized loss minimization, which usually refers to how the concept of SRM is conducted in practice. The optimization problem is then

$$\hat{f} = \arg\min_{h \in \mathcal{H}} \mathcal{L}_{\mathcal{I}}(h) + \rho R(h), \tag{7}$$

where R is a complexity function and ρ is a regularization parameter [10].

In practice, some optimization algorithm is used for solving these optimization problems once an explicit formula is obtained for the objective function after determining the forms of the loss function and complexity function. Error function is the empirical risk $\mathcal{L}_{\mathcal{I}}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h, z_i)$ and thus its formula depends on the selected loss function. In a least-squares problem, the used loss function is squared-loss $\ell(h, z) = (h(x) - y)^2$. Another commonly used loss function is logistic loss $\ell(h, z) = \log(1 + \exp(-yh(x)))$ that is used in logistic regression. Yet another error function is hinge loss $\ell(h, z) = \max\{0, 1 - yh(x)\}$ that is used in support vector machines (SVM). Logistic loss and hinge loss are commonly used in classification tasks as they are convex surrogate loss functions to a zero-one loss. In addition, ERM with log-loss $\ell(\theta, z) = -\log p(z \mid \theta)$ is equivalent to the MLE [10].

The regularization term helps to make the learning rule stable, and it is shown that stable algorithms are less prone to overfitting. A commonly used regularization

function is the ℓ_2 norm of the parameter vector (weights), which is called Tikhonov regularization. This stabilizes the algorithm by penalizing large weights of a linear model. Another commonly used regularization function is the least absolute shrinkage and selection operator (LASSO), which is the ℓ_1 norm of the weights. The use of this kind of regularization leads to a sparse solution, and is thus suitable for feature selection [10]. Enforcing sparsity in the model can be done by using ℓ_0 pseudonorm, which simply counts the number of nonzero components of the vector of parameters. The nonconvexity and discontinuity of the ℓ_0 pseudonorm make the optimization problem NP-hard [25], and hence it is often replaced by the ℓ_1 norm that is its tight convex relaxation [26]. Another way to undertake the problems of nonconvexity and discontinuity is to use the class of polyhedral k-norms, which are defined as the sum of the k largest components of the vector [27; 28].

2.2.3 Optimization Algorithms

Characteristics of the objective function to be optimized by an optimization algorithm depend on the optimization problem and the hypothesis space. In the generative approach, the problems (4) and (6) can sometimes be solved analytically, especially when the hypothesis space consists of simple distributions, e.g. normal, and the prior distribution is a conjugate prior (see e.g. [24] for more information on what they are). However, numerical methods are often used because the models may be more complicated and there may be other things to be taken into account. For example, Expectation Maximization algorithm used in Publication I and to be presented in more detail in the next subsection, can be applied when there are some missing data. There exist many kinds of numerical methods for solving the optimization problems in the discriminative approach. In data science, the optimization algorithms that are commonly used are in the subset of indirect methods, which iteratively search for a solution, improving it at every iteration.

Whether the optimization problem is constrained or not affects sufficient and necessary optimality conditions, which are used to verify the optimality of a solution obtained by an iterative algorithm [29]. All the optimization problems presented in Section 2.2.2 are unconstrained, as even in the SRM, the constraint of the model complexity is taken into account by the regularization term in the objective function. Depending on the interpretation, this kind of consideration is called Lagrange multipliers (see e.g. [30] for an explanation) or penalty function (explained e.g. in [29]).

In addition, characteristics of the objective function affect the methods that are applicable for solving the problem, and what the optimality conditions guarantee. The most important characteristics are convexity and differentiability. From the convexity, it follows that a found solution, which is a local minimum, is also a global minimum. The importance of differentiability is that the main approach when solv-

ing unconstrained problems is gradient descent [30], i.e. many numerical optimization methods utilize gradients to find a direction in which the solution is updated in the following iteration. Another significant characteristic is smoothness of the objective function, because the gradient of a nonsmooth function may not exist for all points. Then the concept of gradient needs to be generalized to a subgradient [31]. Furthermore, the size of the optimization problem has an impact on what algorithms are feasible, as it affects the time and computational complexity of the method. Limited memory bundle method [32] is an optimization algorithm suitable for solving this kind of more complex task, was used as part of the presented method in Publication III and is presented in a separate subsection at the end of this section.

Expectation Maximization Algorithm

Expectation maximization (EM) algorithm is an iterative method, which can be used for example for solving an MLE for a mixture cure model with latent variables. This example is utilized to make the description of the method clearer. The EM-algorithm uses two versions of a likelihood function: one for the observed incomplete data and another for complete data. The incomplete data likelihood function for a mixture cure model and a sample of data of n elements is [14]

$$L^{\text{obs}}(\theta \mid \boldsymbol{z}^{\text{obs}}) = \prod_{i=1}^{n} p(\tau_i \mid \theta)^{1-\delta_i} S\left(\tau_i \mid \theta\right)^{\delta_i}, \tag{8}$$

whereas the complete data likelihood function

$$\begin{aligned} L(\theta \mid \boldsymbol{z}) = \prod_{i=1}^{n} p(\tau_i \mid \theta)^{1-\delta_i} \Big\{ & \left[\mathbb{P}\left(\mathrm{T}^* > \tau_i \mid Y = 1\right) \mathbb{P}\left(Y = 1\right)\right]^{y_i} \\ & \cdot \left[\mathbb{P}\left(\mathrm{T}^* > \tau_i \mid Y = 0\right) \mathbb{P}\left(Y = 0\right)\right]^{1-y_i} \Big\}^{\delta_i}. \end{aligned} \tag{9}$$

The EM-algorithm consists of two steps: in the E-step, conditional expected values of the latent variables are calculated given the observed data, and in the M-step, those values are used in order to be able to maximize the complete data likelihood function (9) instead of the likelihood function for the incomplete data (8). More precisely at an iteration round r, the E-step means taking a conditional expectation of the complete data likelihood $\mathbb{E}_{\theta^{(r)}}\left\{l(\theta) \mid \boldsymbol{z}^{\text{obs}}\right\}$ and finding such $\theta^{(r+1)} \in \Omega$ in the M-step, for which $\mathbb{E}_{\theta^{(r)}}\left\{l(\theta^{(r+1)}) \mid \boldsymbol{z}^{\text{obs}}\right\} > \mathbb{E}_{\theta^{(r)}}\left\{l(\theta) \mid \boldsymbol{z}^{\text{obs}}\right\}$ for all $\theta \in \Omega$. As the optimization problems related to MLE and MAP, (4) and (6), are similar, there is not much difference between the steps of the EM-algorithm for the mixture cure model either. The prior knowledge used in the MAP point-of-view affects the expectation step of the EM-algorithm by adding a term related to the prior distribution. The E-step is then calculating

$$\mathbb{E}_{\theta^{(r)}}\left\{\log p(\theta \mid \boldsymbol{z}) \mid \boldsymbol{z}^{\text{obs}}\right\} = \mathbb{E}_{\theta^{(r)}}\left\{l(\theta) \mid \boldsymbol{z}^{\text{obs}}\right\} + \log p(\theta) \tag{10}$$

and again in the M-step, $\theta^{(r+1)}$ is chosen so that it maximizes (10) over the whole parameter space Ω. When using the EM-algorithm, the two steps are repeated until convergence is achieved. Convergence of the EM-algorithm is measured by comparing the values of (8) at $\theta^{(r+1)}$ and $\theta^{(r)}$, and is considered to be achieved once the difference between the values is small enough [33].

Limited Memory Bundle Method

Limited memory bundle method (LMBM) is an iterative optimization algorithm, which only expects the optimization problem to be unconstrained and the objective function to be locally Lipschitz continuous, and in particular, does not require either the convexity or the differentiability of the objective function. It is specifically suitable for solving large problems as its computational complexity is only linearly dependent on the number of variables [32].

Main steps of the LMBM are direction finding, line search and aggregation of subgradients. An aggregate subgradient is used for direction finding instead of the usual gradient. A two-step line search procedure is used for determining the step to be taken into the determined direction, because further information is needed about the goodness of the search direction as it is possible that it is not a descent direction in the nonsmooth optimization. The step is either a serious step or a null step, which mean, respectively, that the direction is descent one and the solution is updated in that direction, or the solution remains the same as in the previous iteration, but additional information about the objective function is provided through an auxiliary point and subgradient. The aggregation procedure is conducted only after a null step, and it uses three subgradients and two locality measures from the earlier iterations. Another term affecting the search direction is a correction matrix that accumulates information about the previous subgradients. The steps of direction finding, correction, line search and aggregation are iterated until a small enough value is achieved for a stopping parameter, which is calculated as a function of aggregate subgradient, search direction and locality measure from the current iteration [32].

2.2.4 Learning Algorithms

Learning algorithm is a procedure that combines the three components presented in the previous sections into one algorithm that returns the optimal hypothesis given the data and the model. They are implementations that utilize an optimization algorithm to solve an optimization problem with a given set of possible hypotheses so that the final hypothesis is the best at the given task for the given sample of data. In statistical inference, the learning procedure is not typically referred to as a learning algorithm, and thus, the methods presented in this section are all for the discriminative approach. Even further, the methods presented in this section are limited to the ones which were

used in Publications II and III. Contents of the three components of the learning algorithms are given in Table 2 for the exact versions that were used.

Table 2. Examples of learning algorithms based on what was used in the publications. The algorithm presented in Publication III, LMBMKronℓ_0LS, is included to facilitate comparison with the corresponding baseline methods. Definitions of the acronyms used as a part of an algorithm name or only in this table: CART refers to classification and regression trees, CG to conjugate gradient, RLS to regularized least-squares, RBF to radial basis function, and LS to least-squares.

Learning algorithm	Hypothesis space	Optimization problem	Optimization algorithm
Linear regularized least-squares [34]	Linear models	SRM with squared loss and ℓ_2 norm as regularizer	Singular value decomposition based approach [35]
Logistic regression [36]	Linear models transformed by Sigmoid function	SRM with log-loss and ℓ_2 norm as regularizer	Liblinear [37]
Random forest [36; 38]	Tree-based	ERM with Gini impurity	CART [39]
CGKronRLS [34]	RKHS with Gaussian RBF kernel	SRM with squared loss and ℓ_2 norm as regularizer	Conjugate gradient with 'vec-trick' [40]
KronSVM [34]	RKHS with Gaussian RBF kernel	SRM with squared hinge loss and ℓ_2 norm as regularizer	Quasi-minimal residual method [41] with 'vec-trick' [40]
LMBMKronℓ_0LS [42]	RKHS with Gaussian RBF kernel	SRM with squared loss and ℓ_0 pseudonorm as regularizer	LMBM [32] with generalized 'vec-trick' [43]

In addition to learning algorithms for regression or classification tasks, there are algorithms that produce continuous outputs that can be used for ranking the data, but that are not labels in the same manner as the continuous predictions in a regression task. This kind of algorithms are called scoring algorithms [44]. The continuous scores can then be transformed into binary labels determined by a threshold value.

Binarization of the scores is highlighted by using notation

$$\kappa_\gamma(i) = \begin{cases} 1 & \text{if } \hat{f}(x_i) > \gamma \text{ and} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

for this kind of a classifier, where γ denotes the used threshold value. For example, logistic regression is such a classifier, where the continuous scores are obtained by transforming the outputs of a linear model into probabilities of belonging to a certain class. Similarly, random forest can produce probabilities of belonging to a certain class, whereas in SVM, the scores would be distances of the data points from the separating halfspace [10].

2.3 Evaluation Phase

In order to be able to reliably use a model or method, its quality needs to be evaluated. There are two types of evaluation in the predictive data science process, as was visualized in Figure 3: model selection and generalization performance. The focus in model selection is on comparing different models and selecting the one that fits the data the best. In general, the realizability assumption is not fulfilled ever as all models are simplifications of reality. However, some models can still be better than others in the given tasks, and thus examples of methods for model selection are presented in Section 2.3.1. As was seen in the Figure 3, the model selection can be done on the same data that is used for inferring the candidate hypothesis, or on a separate validation data when the goal is to select a hypothesis that is the best among the candidate hypotheses at generalization. Similarly, the focus of generalization performance evaluation is measuring how well the final hypothesis generalizes to new data. Thus, yet another separate data set is needed for testing. Examples of performance measures for estimating the generalization ability are presented in Section 2.3.2.

The amount of available data might become a problem when the evaluation is to be done on data that was not seen while learning the hypothesis. If there were plenty of data, simple hold-out could be used to split the sample into training, validation and test sets so that $\mathcal{I} = \mathcal{I}_{tr} \coprod \mathcal{I}_{val} \coprod \mathcal{I}_{test}$, where the subscript notation denotes a subset of $\mathcal{I}$ and $\coprod$ highlights that the union is taken over disjoint sets, in order to have separate data sets for model selection and model assessment, but as the amount of observed data is usually limited, resampling methods are often used to make better use of the available data. The most important resampling methods are mentioned in Section 2.3.3, although the focus is on cross-validation, which was a main component of the presented method in Publication II.

2.3.1 Model Selection

Learning algorithm returns an optimal hypothesis among the given hypothesis space, but there may be several models from which the most suitable one should be selected. Then the learning procedure is executed for the different models, and model selection is used to determine the optimal final hypothesis. Thus, model selection can refer both to selecting between several hypothesis spaces or learning algorithms and to tuning some hyperparameters of a learning algorithm.

The data that are used for model selection depend on the goal of the used data science approach. In the generative approach, the model is a guess of the distribution from which the data are sampled, and its quality is measured in the sense of how well the optimal hypothesis fits the data. This is exactly the definition of a likelihood function, which returns the likelihood of having observed the given sample of data under the model. Hence, the likelihood function can be used as a component of different criteria that can be used to select the optimal model. For example, **Akaike information criterion (AIC)** [45] is one of these kinds of criteria. Its value is calculated with

$$\mathrm{AIC} = 2\,|\theta| - \log L(\hat{\theta}),$$

where $|\theta|$ denotes the number of parameters of the model. Thus, the complexity of the model is also considered, and the more complex models are penalized for having more parameters to be optimized. The values of these kinds of criteria computed for several models can be compared, and the one producing the lowest value can be selected as the final hypothesis.

In contrast, in the discriminative approach, where the goal is to find a hypothesis that is good at generalizing to new data, the model selection also needs to be done on data that are separate from the training data. Then, separate validation data are used. When it comes to this approach, the model selection may be selecting between different learning algorithms or different models obtained by tuning some hyperparameters of a learning algorithm. The selection can be conducted based on the same measures that are suitable also for evaluating the generalization performance.

2.3.2 Generalization Ability

Generalization error measures the model's ability to predict on new data. Generalization performance measures can be based on the predicted values, which measure how close the predicted values are to the true values, or on the order of the predictions, which measure whether the order of data points is preserved during the prediction task. For the latter goal, continuous predictions can be used directly, but in a classification task, continuous scores are needed instead of the class labels.

When it comes to a classification task, measures based on a confusion matrix are commonly used. Elements of a confusion matrix can be defined for a classifier κ_γ by

$$\sum_{i \in \mathcal{I}_{test}} \left[(y_i = c_t) \wedge (\kappa_\gamma(i) = c_p) \right],$$

where c_t and c_p have all possible classes at turn [44]. In a case of binary classification, it is a 2×2 matrix, whose elements are the numbers of correctly/falsely predicted positives and correctly/falsely predicted negatives. This is usually denoted for a deterministic classifier with a symmetric loss, but in Table 3, the notations related to a scoring algorithm are used, and the subsets of the test data having positive or negative labels are denoted by $\mathcal{I}_+$ and $\mathcal{I}_-$, respectively. Several performance measures can be derived from a confusion matrix: accuracy, precision, recall, sensitivity, specificity, true positive/negative rates and false positive/negative rates etc.

Table 3. A confusion matrix.

	Predicted positive $\kappa_\gamma = 1$	Predicted negative $\kappa_\gamma = 0$
Positive label $\mathcal{I}_+$	True positives	False negatives
Negative label $\mathcal{I}_-$	False positives	True negatives

The measures derived from the confusion matrix can be presented as conditional probabilities. Sensitivity, true positive rate (TPR) and recall are all equal to $\mathbb{P}\left(\kappa_\gamma(i) = 1 \mid i \in \mathcal{I}_+\right)$, which is the probability for correctly predicting a positive datum as positive. Similarly, $\mathbb{P}\left(\kappa_\gamma(i) = 0 \mid i \in \mathcal{I}_-\right)$ is the probability for correctly predicting a negative datum as negative, also called specificity or true negative rate. Hence, the false positive rate (FPR) is the complement of specificity. False positive and false negative rates are also called type I and II errors, in the respective order [46]. All of these measures have a single-class focus as they focus either only on the positive data points or only on the negative data points. On the other hand, accuracy $\mathbb{P}\left(\kappa_\gamma(i) = y_i\right)$, which is the most commonly used performance measure in classification tasks, measures only the portion of correctly labelled data points, no matter what their true classes are [44].

As the elements of a confusion matrix are based on predictions made by a deterministic algorithm and using symmetric loss, there are limitations in the usage of these measures. For example, if the data are highly imbalanced, a high accuracy can be obtained by always predicting the majority class. In addition, as was said, the measures derived from a confusion matrix do not account for possible asymmetric misclassification costs. Nevertheless, these can be taken into account by changing the threshold value in (11). Then, a confusion matrix becomes a tensor as there would be another dimension for the threshold value [44], which, however, is not easily interpretable. Another way to deal with this is to use such ranking-based performance measures that use the scores directly or take into account different values for γ. From

now on, the $\hat{f}$ is denoted as a function of an index to simplify the notations and the index set for the used training data is marked in subscript when necessary: for example $\hat{f}_{\mathcal{I}_{tr}}(i)$ denotes the value of the final hypothesis inferred from training data $\mathcal{I}_{tr}$ evaluated at a test data input x_i corresponding to index $i \in \mathcal{I}_{test}$.

Area under the ROC curve (AUC) is a ranking-based summary statistic that measures the probability of a randomly selected positive datum to be ranked higher than a randomly selected negative datum:

$$\mathbb{P}\left(\hat{f}(i) < \hat{f}(j) \mid i \in \mathcal{I}_-, j \in \mathcal{I}_+\right). \tag{12}$$

In order to estimate this probability, all pairs of a positive and a negative datum need to be considered. As a generalization of AUC, **concordance index (C-index)** is another ranking-based performance measure producing a point estimate. The use of the concordance index is not limited to only binary data sets, as it only compares whether the predictions of two data points are in accordance with the order of their true labels. Then the concordance probability is $\mathbb{P}\left(\hat{f}(i) < \hat{f}(j) \mid y_i < y_j\right)$, which is equal to (12) if $\mathcal{Y} = \{0, 1\}$ [47].

Graphical measures can provide more information about the effect of the threshold. For example, **Receiver Operating Characteristic (ROC) curve** is a ranking-based graphical measure that takes into account the effect of the threshold. A ROC curve is obtained by calculating the TPR and FPR for all possible threshold values and drawing the obtained points in a unit square called a ROC space. An example of three ROC curves is presented in Figure 4 to demonstrate possible shapes of the curve. The closer the curve lies to the left top corner, the better the classifier is. If the corner is reached, there exists a threshold that leads to a perfect classification of the test data points. In contrast, the curve lies on the diagonal for a random classifier. ROC curve uses two distinct single-class focus measures rather than combining those into a single multiclass focus metric. This is another reason why ROC curve analysis is suitable for studying the class imbalance [48]. In practise, a ROC curve can be drawn by ranking the data from the smallest to the greatest according to the predicted scores, $\hat{f}(i_1) \leq \hat{f}(i_2) \leq \cdots \leq \hat{f}(i_{|\mathcal{I}_{test}|})$, calculating the predicted labels with (11) for every possible $\gamma \in (-\infty, \infty)$, and using those predictions for calculating the conditional probabilities. The starting point of a ROC curve, (0,0), is obtained when every datum is predicted to belong in the negative class, i.e. when $\hat{f}(i_{|\mathcal{I}_{test}|}) < \gamma$. Similarly, the endpoint of it, (1,1), is obtained when $\gamma < \hat{f}(i_1)$, implying that all data points are predicted to belong in the positive class. Otherwise, the TPR and FPR are both within the range [0, 1].

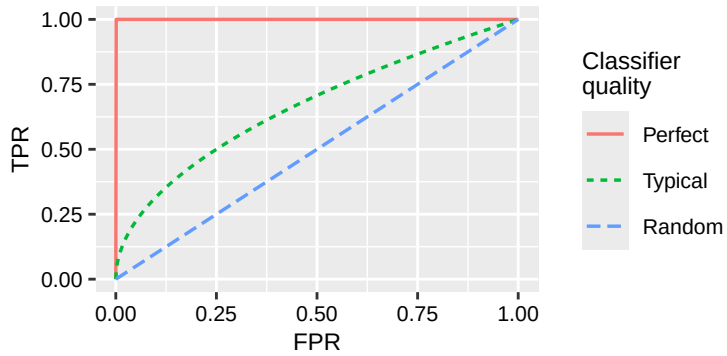

Figure 4. An example illustrating the ROC curves of three classifiers with different ranking ability.

2.3.3 Resampling Methods

Resampling methods provide means for versatile use of a data sample of limited size. Their usage is justified by trying to balance between having enough data for learning an unbiased hypothesis and having enough data for estimating the algorithm's generalization ability so that the variance of the error estimates would not depend that much on the random variation in the test data [44]. Resampling methods can be divided into subcategories of multiple and simple resampling based on whether each data point is used for testing possibly multiple times or only once. Multiple resampling is further divided into random subsampling, bootstrapping, randomization and repeated cross-validation. Simple resampling, then again, is further divided into resubstitution and cross-validation [48]. Further explanation about cross-validation will be given next, as the only resampling methods used in the publications are the different versions of cross-validation that were used in Publication II.

In Figure 3, the data were visualized as being split into separate training and test sets, but sometimes there are insufficient data for both training and evaluation, if this kind of simple hold-out was used. This problem can be avoided by repeatedly separating a smaller part of the data for evaluation and using the rest for learning the hypothesis, and then using an average of the performance estimates obtained from the repetitions as a final estimate. In cross-validation, the idea is to split the sample into several folds of approximately equal size, and iterate through the folds so that each fold is used once as a separate test set while the other folds are used for training a model. This ensures a more structured way to split the data compared to random repeated hold-out. When using cross-validation, there are as many final hypotheses as there are folds, because the final hypothesis depends on the training data, which is slightly different at each iteration. Performance estimates can then be either calculated for each test fold separately and averaged, or the predictions made

by different final hypotheses can be pooled, and an estimate is calculated for all the pooled predictions at once [49].

Leave-one-out (LOO) and k-fold cross-validation are the most commonly used cross-validation methods [44]. In LOO, each fold contains only a datum, i.e. the folds are $\{1\}, \{2\}, \ldots, \{n\}$, and thus the test sets are independent from each other. On the other hand, the training sets $\mathcal{I} \setminus \{1\}, \mathcal{I} \setminus \{2\}, \ldots, \mathcal{I} \setminus \{n\}$ are almost completely overlapping, which causes that the final hypotheses

$$\hat{f}_{\mathcal{I}\setminus\{1\}}, \hat{f}_{\mathcal{I}\setminus\{2\}}, \ldots, \hat{f}_{\mathcal{I}\setminus\{n\}}$$

from the cross-validation rounds are not necessarily independent, and affects the bias of the error estimates [44]. In k-fold cross-validation, the sample is split into k folds so that $\mathcal{I} = \coprod_{\nu=1}^{k} \mathcal{I}_\nu$ and $|\mathcal{I}_\nu| \approxeq \frac{n}{k}$ for all ν. Again, the test sets are independent of each other, and this time there is more variation also in the training sets $\mathcal{I} \setminus \mathcal{I}_\nu$, $\nu \in \{1, 2, \ldots, k\}$ as more data are left out for testing.

In contrast, in leave-pair-out (LPO) cross-validation [50], the folds are not independent from each other, because each negative datum $i \in \mathcal{I}_-$ is paired with every positive datum $j \in \mathcal{I}_+$ and every such $\{i, j\}$ pair is once left out for testing. This cross-validation approach can be used when the goal is to estimate a learning algorithm's ability to find a hypothesis whose generalization performance is good with respect to ranking-based summary statistics, e.g. AUC in (12). Furthermore, it has been shown to produce an almost unbiased estimate of the AUC [50], when it is calculated as

$$\widehat{\mathrm{AUC}}\left(\hat{f}_{\mathcal{I}}\right) = \frac{1}{\mid \mathcal{I}_+ \mid\mid \mathcal{I}_- \mid} \sum_{i \in \mathcal{I}_-} \sum_{j \in \mathcal{I}_+} H\left(\hat{f}_{\mathcal{I}\setminus\{i,j\}}(j) - \hat{f}_{\mathcal{I}\setminus\{i,j\}}(i)\right), \qquad (13)$$

where H is the Heaviside function, whose value at argument a is

$$H(a) = \begin{cases} 1, & \text{when } a > 0 \\ \frac{1}{2}, & \text{when } a = 0 \\ 0, & \text{when } a < 0. \end{cases}$$

As the concordance index is a generalization of AUC and uses a similar formula for calculating the estimate, LPO also produces an almost unbiased estimate of it. A benefit of using LPO is that it enables using maximal amount of data for training while comparing only predictions that are made with one prediction function, which has been shown to lead to more reliable results than with LOO and k-fold cross-validations [50]. However, there is also a limitation in its usage: it is not suitable to be used together with all ranking-based performance measures, because it does not produce a full ranking of data. This problem has been tackled by presenting tournament leave-pair-out (TLPO) cross-validation [51], which expands the idea of

LPO to all possible pairs. Then

$$\widehat{\mathrm{AUC}}\left(\hat{f}_{\mathcal{I}}\right) = \binom{n}{2}^{-1} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}: y_j > y_i} H\left(\hat{f}_{\mathcal{I}\setminus\{i,j\}}(j) - \hat{f}_{\mathcal{I}\setminus\{i,j\}}(i)\right), \qquad (14)$$

and a ranking of data can be determined by tournament scores

$$Score\,(i) = \sum_{j \in \mathcal{I}\setminus\{i\}} H\left(\hat{f}_{\mathcal{I}\setminus\{i,j\}}(j) - \hat{f}_{\mathcal{I}\setminus\{i,j\}}(i)\right),$$

which denotes for a datum i, how many times it was predicted to have a higher value than the other sample units.

A downside of using cross-validation methods is that it is time-consuming to repeat the learning process many times. When using cross-validation, as many hypotheses are learned as there are folds, as can be seen e.g. in (13) and (14) for calculating the AUC. The number of hypotheses to be learned is n with LOO, k with k-fold, $\mid \mathcal{I}_+ \mid\mid \mathcal{I}_- \mid$ with LPO, and $\binom{n}{2}$ with TLPO cross-validations. Hence, time-complexity is also an important factor to be taken into account when considering the suitability of using a cross-validation method in a specific case.

Quicksort

In Publication II, Quicksort was combined with LPO cross-validation to decrease the time complexity for producing a full ranking of data from the quadratic time of the TLPO cross-validation. Quicksort is a recursive sorting algorithm that is based on the divide-and-conquer paradigm. In that a pivot-element i_{pivot} is randomly selected among the points $\mathcal{I}$ to be ordered. Then, other points in the current set are compared to the pivot element and divided into two groups based on whether they are smaller or greater than the pivot element. Then the same is repeated in the smaller subsets of indices until the order is determined [13]. In the worst case, the pivot element is always the smallest or the greatest among the current set, causing all the compared data points to end up in the same subset. In contrast, in the best scenario, the pivot element divides the data points equally between two subsets. Thus, the worst-case runtime for quicksort is $\Theta\left(n^2\right)$, whereas its expected runtime is $\Theta\left(n \log n\right)$ [13].

3 Method Development

According to Blei and Smyth [3], developing tools and methods is a main priority for data science. However, there does not seem to be a common agreement on the best practices of conducting data science in general, or method development in particular [52; 2]. Different data science life-cycle methodologies (e.g. [53; 54; 55; 56; 57], and see especially [58] for a comprehensive review), frameworks (e.g. [59; 60; 61; 3; 2]) and checklists (e.g. [62; 63; 64; 65; 66; 67; 68; 69; 70; 71], and see especially [72] for a comprehensive review of existing tools) have been created to formalize the data science pipeline in varying contexts and implementation of the methods in different applications. The life-cycle methodologies consist of abstract steps that can be followed to conduct data science, and checklists are even more practical application-specific lists of actions, whereas the frameworks have a more philosophical perspective as also ethical questions and the communication point-of-view are taken into account in addition to the computational, statistical and scientific perspectives [2] that align with the steps of the life-cycle methodologies. In the scope of predictive data science, the most important steps are understanding the domain and data in order to frame the scientific questions, and using statistical and computational tools to formulate, learn and evaluate a model. Method development is a process of creating new methods to solve scientific problems in a field of application, or more commonly, improving or modifying existing methods to be better or suitable for solving the problems. To formalize this process, Aliferis and Simon [11] have presented multiple quality aspects of data science methods, and a work-flow to describe the steps of the development process of new methods.

3.1 Quality Aspects

There are different quality aspects of data science methods, which are also possible areas for improvement and criteria for comparing methods. According to Aliferis and Simon [11], they are

- representation power,
- transparency,
- soundness,

- completeness,
- computational complexity for learning or using a model,
- other cost functions,
- space complexity for learning, storing or using a model,
- sample complexity, learning curves and power-sample requirements, and
- probability and decision theory consistency.

Briefly, in other words: a good method is suitable for solving all relevant problems, is not a 'black box', produces correct solutions for the whole problem space, can be used on a reasonable amount of time, memory, data, and other resources, and is theoretically justified from the points-of-view of probability and decision theory. In addition to the theoretical justification, soundness and completeness are more like theoretical properties that guarantee that the method is reliable to use. In contrast, the other quality aspects are more relevant to the usability of the method. Representation power, transparency, and time and memory complexities are the most relevant quality aspects when it comes to the research studies presented in Chapter 4. Thus, further description of them is given in the context of the components of predictive data science.

Representation power of a method depends on the representativeness of the observed data, components of the learning process and the selected performance evaluation measure. It is clear that the final conclusions cannot be trusted if the questions are not properly defined, the data are not a representative sample of the interested population, and the selected methods are not suitable for answering the questions [2]. Quality of the data greatly impacts the quality of models that can be inferred from it: a good model cannot be learned from inadequate data, or vice versa, garbage in, garbage out. Thus, the observed data need to be a representative sample of the population that is being modelled [73]. Representation power of a learning algorithm answers a question of what type of relationships the method can represent [11]. This means the suitability of the selected hypothesis space, and whether the optimization problem and algorithm are selected in accordance with the goal of the problem. The representation power of a hypothesis space is connected to its complexity, and the complexity of the model has a positive relationship with its ability to model complex data. However, the more flexible the model, the higher the risk of learning also the noise in the training data, and the lower the possibility of understanding, interpreting and explaining the model. Finally, the choice of performance measure also affects the representation power of the whole process: it needs to be selected so that it measures meaningful and reasonable performance criteria [11].

Transparency of a method means that its steps are possible to understand and easily interpretable by humans [74]. In addition, there should be a clear relation

between the method semantics and real-world entities for transparent methods [11]. Only the simplest models are transparent in that sense, but for example, regularization (see the end of Section 2.2.2) can be used to penalize more complex models and thus increase the explainability of the final model. In addition, there exist several methods or tools to increase the explainability of the black-box methods [74], but the downside of using them is that they simplify the model's complexity too much.

The computational and space complexities are theoretical characteristics of methods, and denote the asymptotic or exact amounts of time and memory needed for using the methods. They are usually written as functions of sample size, and need to be determined separately for learning, using or storing a model in predictive data science [11]. These aspects provide another point-of-view for comparing methods and selecting possible methods in addition to comparing only the prediction performances of the methods.

Altogether, all of these quality aspects can be used for validating and comparing methods from several points-of-view. In addition, they can also be used for justifying why a presented new method would be better than the previously existing ones. They also form the basis for possible aspects of method development while refining and improving existing methods during the process of developing methods.

3.2 Process

A precise description of a method development process has been presented by Aliferis and Simon [11]. They have described it as a work-flow starting from a rigorous problem definition and theoretical analysis of the problem, then moving on to theoretically analysing the presented method for solving the problem, and finally empirically testing the method both in controlled conditions and with real-life data. The work-flow is highly non-linear, as returning to an earlier step for some refining may be necessary [11]. Based on the work-flow, the data science method development process is described in the following subsections to have three steps: defining and analysing the problem, conducting theoretical analysis on the presented method, and empirically validating the method. Furthermore, it is important to communicate the results at the end of the method development process, as is highlighted in the data science life-cycle methodologies, and frameworks and checklists for conducting data science, for example, by Lones [52]. Reporting the results is related to the human perspective of data science, which is what Baillie et al. [2] used as an unifying perspective that brings the project all together. In addition to that reporting the method and results transparently enforces open science, it also enables others to be able to verify that the results are sound and reliable. Reproducibility, replicability, robustness and generalizability are different aspects of reliability and repeatability [63] that provide useful information on what can actually be said about the results and what conclusions can be drawn based on them.

3.2.1 Defining and Analysing the Problem

The process begins with properly defining the problem that is to be addressed by the new method. Scientific thinking enables defining and iteratively refining the problem based on the available information, whereas statistical perspective provides means for analysing the problem [2]. A precise and mathematical definition enables analysing the problem theoretically and proving whether it can or cannot be solved within given performance requirements and with the available resources. Theoretical analysis of the problem provides the means to select the methods that could be used for solving it.

Once the problem is defined and the population of interest is determined, a sample of data is needed to represent the population and be used together with the selected methods. Benchmark data sets are such that are publicly available and commonly used to validate methods used in a specific field, though it is good to keep in mind that using these might bias the methods towards being good on these data but not on other real-world data sets [52]. Another way to obtain the sample is secondary use of existing data [59], i.e. data that were collected for another purpose but can be used for this task as well. However, the secondary use of data comes with a risk of it being incomplete, noisy or non-representative [59] as well as with ethical concerns.

3.2.2 Theoretical Analysis of the Method

Theoretical analysis and empirical validation together provide information on the suitable methods and their properties for the task. Theoretical analysis is needed in addition to the empirical testing in order to avoid having to do an enormous amount of empirical studies, to have the necessary information for planning the empirical studies, and to have a broader analysis than what can be obtained by using benchmark data sets [11]. In addition, it is possible to confront some unexpected consequences in the execution or interpretation of the method, if computational considerations are not taken into account before implementing a method [2]. Based on the results of the theoretical analysis of the method, it can be made more suitable, and the refinements and extensions determine the practical utility of it [11].

Aliferis and Simon [11] have listed different categories of possible types of refinement:

1. Optimizing the need for resources.
2. Changing the way the computation is executed, e.g. by parallelization.
3. Relaxing the necessary or sufficient conditions.
4. Enlarging the set of possible methods for a task or applications for a method, i.e. generalized frameworks.

5. Making the method more easily understood by increasing its explainability, clarity or transparency.

Within the context of the research studies included in this thesis, the most relevant refinement categories are the performance optimization and the generalized frameworks.

Performance optimization refers to optimizing the method's requirements of any kind of resources, e.g. improving the computational, space or sample complexities of the method, or optimizing some other criterion determined by other cost functions [11]. An implementation of the method affects its efficiency greatly, as is seen in the runtime analysis in the Publication II, where it was faster to have a larger number of predictions at once than to obtain them in several smaller parts with the implementation for RLS. Nevertheless, the modifications in the refinement category 2 on the implementation of the method usually lead to better efficiency by dividing the execution into smaller problems.

The generalized frameworks, then again, mean having a core method included in a family of similar methods, justifying its belonging to there and establishing rules that guarantee common properties within the family of methods. This kind of generalization enables that the core method can be modified to be suitable to solve a slightly different problem, or used as a basis for variations, which inherit some properties of the core method [11].

All the algorithm refinement categories form a versatile list of possible aspects of what could be improved in existing methods over time, and they affect the quality aspects of a method (see Section 3.1). However, the analysis of theoretical properties of a method only demonstrates the theoretical possibility of how a method works, but cannot guarantee its usability with real-life data. That is why it is important to combine the theoretical analysis with empirical testing [11].

3.2.3 Empirical Validation of the Method

Empirical validation of a method starts with testing it in controlled conditions, which means testing how different characteristics of data affect the performance of the method. In practice, this means testing for example different sample sizes, amount of noise in the data, strength of the signal in the data or the dimensionality of the data. This is also the place for testing how the results are affected if the assumptions for the data are violated to a varying degree. Depending on the aim of the controlled conditions tests, the data can be simulated, i.e. generated based on a mathematical model, re-simulated, i.e. generated from an encoded distribution learned from a real-life distribution, or shuffled, which is similar to the idea of using data in the permutation test [75] and means permuting the data so that on average there is no signal to be learned. Additionally, other statistical tests can be used alongside other evaluations to verify that the real data fulfils the assumptions of the data for the method. It is

important to verify the assumptions on the data, because it is possible not to be as anticipated, which onward has an effect on the results obtained by the method [11].

The other part of the empirical validation of a method is to test it with real-world data sets. Aliferis and Simon [11] split this part into two steps: in the first step, the labels are known, whereas in the second they are not known but instead are assumed to be obtained later. The empirical testing is usually done by using available benchmark data sets and comparing the performance of the presented method to several state-of-the-art alternative methods. However, it is important to remember that using benchmark data sets for comparing methods has its risks of the best method over-fitting the test set and not being any better at generalizing than the other methods [52; 76].

Yet another thing to be taken into account when empirically testing a method and comparing it to others is to use such baseline methods that are a meaningful basis for comparison. Naïve baselines are the simplest methods used for demonstrating that the problem is not trivial. More generally, the baseline methods are often simpler than the presented new method, and are used to justify the use of the more complex model [52]. Sometimes the simpler methods can be even competitive with the more complex ones (see e.g. [18], where this kind of side finding was observed). This, though, may be a consequence of not having a proper method to measure what is actually wanted to be measured, and reminds of the necessity for carefully choosing the performance metrics so that they are suitable for the case and e.g. take into account the class imbalances or the type of the errors made, if there are unequal costs for different errors. That is why it is recommended to use multiple metrics in the empirical validation phase and report them in the end [52].

4 Research Studies and Results

Different methods were developed and presented in the research studies included in this thesis, targeting different kinds of problems in various domains. The types of method development done in the studies are presented in Table 4. The development processes have somewhat followed the work-flow described in Chapter 3, and hence the research studies are discussed accordingly in the following sections.

Table 4. Summary of the methods developed in the research studies.

Publ.	Development tactics	Impacted quality aspects	Development category
I	A mixture cure model was formulated for the domain of mobile games for profitability prediction.	Representation power of survival analysis mixture cure models was expanded to this domain.	Generalized frameworks.
II	Quicksort was combined with LPO cross-validation to avoid the need for running a complete tournament to obtain a full ranking of data.	Computational and space complexities of a cross-validation method for ROC curve analysis were decreased.	Performance optimization.
III	The optimization component in the CGKronRLS learning algorithm was modified to enforce sparsity in the model.	Considering sparsity in addition to the prediction performance in the main objective impacted other cost functions, decreased the model complexity and increased the model transparency.	Generalized frameworks.

4.1 Bayesian Inference for Predicting the Monetization Percentage in Free-to-Play Games

This study serves as an example of conducting method development by formulating a commonly used model in a new domain and deriving the forms of the necessary formulas for this particular case with relevant distributions and assumptions. The development type falls in the category of generalized frameworks, when considering from that category the aspect of modifying a core method so that it is suitable for solving a slightly different problem than for which the method is usually used. It is relative, what is considered as a slightly different problem, but in this case, it meant that the data had similar characteristics despite being collected from a completely different application than where these kinds of models are generally used. Further details are described in the following sections about the problem that was solved, the method developed for solving the problem and how the method was empirically validated.

The problem

Free-to-play games have become to account for the majority of all games, which has led to the revenue models evolving. In free-to-play games, the money comes from advertisements, premium upgrades and in-app purchases instead of game purchases and subscriptions. Then it is important to know as early as possible, after publishing the game, how many players are expected to monetize in order to know whether it is profitable to continue developing the game.

The prediction task is then in the field of semi-supervised learning, because the labels representing whether or not the individuals will monetize are known for only some players, i.e. those who have already monetized. This is demonstrated by a simulated example in Figure 5, where the endpoint characters denote whether the individuals were detected to monetize or censored due to the limited follow-up. Additionally, the monetization indicator is presented by colours in the figure. This is the information that is needed for predicting the profitability of the game.

The method

The presented model is based on survival analysis methods, modelling the players as a mixture cure model and finding estimates of the model parameters by MLE or MAP, if there exists prior information about the distribution of the monetization percentage. To recap from Section 2.1.1, there are three random variables used for describing the situation: time, censoring indicator and label that now corresponds to the monetizing indicator. Monetization can be expressed as a survival analysis model because the data gathered from a new game shortly after publishing it are

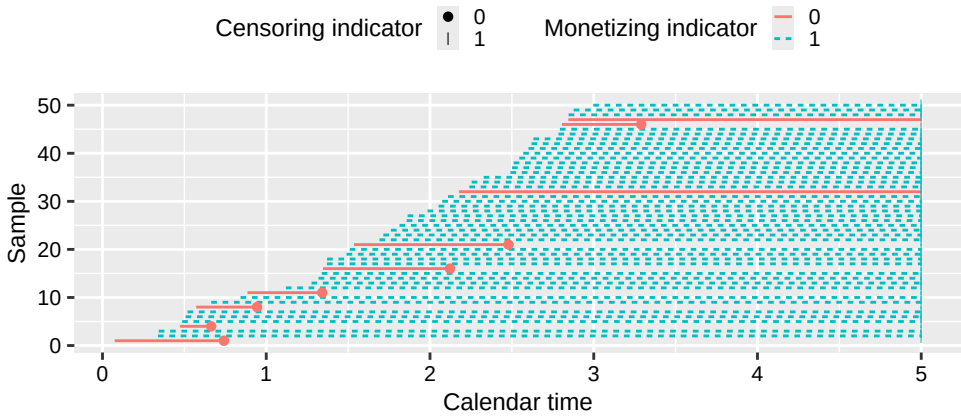

Figure 5. Simulated data example with 50 players: 9 monetizing and 41 unmonetizing. The data were generated with parameters $(\pi, \lambda) = (0.1, 1.0)$. Figure adapted (subfigure resized and font changed) and caption from Publication I [77], ©2020 IEEE.

time-to-event data with limited follow-ups, and the event of interest is to make the first purchase. The standard assumption of survival analysis that everyone eventually has the event does not apply in free-to-play games since many players never seem to purchase anything. Hence, the population is a mixture of monetizing and unmonetizing players. In addition, prior information can be used according to the principles of Bayesian inference for the monetization percentage to ease the prediction task when there is not much data gathered yet.

Now, the hypothesis space is the unconditional survival function of a mixture cure model (2), where an assumption of the distribution for the time variable will be substituted. Then the exact form of the hypothesis is

$$h(\tau) = \pi e^{-\lambda\tau} + 1 - \pi,$$

when the time is assumed to follow exponential distribution $\mathrm{Exp}\,(\lambda)$. Thus, the parameters to be optimized are π and λ, which denote the monetization percentage and conversion rate. The model is then fit through the MLE or MAP optimization problems (4) or (6), respectively, depending on whether there is prior information available.

Empirical validation

Validity of the presented model was studied by numerical experiments conducted on two generated data sets, a relatively small game data set gathered from several versions of a free-to-play mobile game and a larger real-world data set (CDNOW data set [78]) having similar type of data. The generated data sets were used to

measure how much the amount of data or the true monetization percentage affect the estimates of the monetization percentage obtained through the model learning process, and the real-world data sets were used to create censored data sets that replicate the actual version user test.

Methods

The model was fit by the EM algorithm because of the latent variable in the model. In addition, Q–Q-plots and AIC were used for verifying that the model assumptions are fulfilled with the real-world data: Q–Q plots for verifying that $\mathrm{T}^* \sim \mathrm{Exp}\,(\lambda)$, and AIC for model selection between a survival analysis model and a mixture cure model. Drawing Q–Q-plots is a method of exploratory data science that compares the sample quantiles to the theoretical quantiles of the distribution the sample is assumed to follow.

Controlled conditions data

The experiments with generated data were run as Monte Carlo simulations with a thousand repetitions in order to numerically approximate the distribution of the estimate obtained by the method. In the first experiment, the aim was to evaluate the effects of sample size and follow-up time on the estimate of the monetization percentage, because they affect the number of events that have occurred. The generated data sets had 100, 500, 1 000, and 5 000 data points and 10, 50, 100, and 500, respectively, monetizing individuals as the true monetization percentage was $\pi = 0.10$. Several follow-up times were determined so that there was 25 %, 50 %, 75 % or 100 % probability for the monetizing players to purchase before censoring. Additionally, the impact of the choice of a prior distribution was taken into account by selecting two distributions with different means and variances: $\mathrm{Beta}\,(2, 10)$ and $\mathrm{Beta}\,(10, 82)$.

In the second simulation experiment, the goal was to test the effect of the true monetization percentage on the estimate. It was tested with values 0.01, 0.05, 0.1, 0.5 and 0.75. Censoring times were defined the same way as earlier, but with $\{10\,\%, 20\,\%, \ldots, 100\,\%\}$ probabilities to purchase before censoring and in this experiment, the sample size varied from 100 to 1 000 in steps of 100. Prior distributions $\mathrm{Beta}\,(2, 100)$, $\mathrm{Beta}\,(2, 20)$, $\mathrm{Beta}\,(2, 10)$, $\mathrm{Beta}\,(2, 2)$, and $\mathrm{Beta}\left(2, \frac{4}{3}\right)$ were selected so that their mean values vary approximately in the range of the true monetization percentages.

Real-world data

The game data consisted of data gathered from six different versions of the game. The number of players varied between the versions from a few hundred to over 2 000. The number of monetizing players was small in all versions, varying from 6 to 35. Hence, the true monetization percentages varied in the range 0.004–0.019. The date of starting to play the game represented the time of entering the study, and the time from that to the date of the first purchase was considered as the event time. Different data collection dates were simulated by creating censored data sets by considering only the data from the first $1, 2, \ldots$ days since the beginning of the study. Maximal follow-up was the latest date in the whole data set.

The CDNOW data set was used similarly to validate the model on a data set with a larger sample size (23 570) and greater true monetization percentage (approximately 50 %), i.e. with more information. It contained information on the purchase history of 23 570 individuals in an online retail shop. Similar characteristics can be derived than from the mobile game data, with the difference that now the time of entering the study is the date of the first purchase, and the event of interest is whether the customers return to the shop and make another purchase. The different follow-up times were created in the same way as for the game data, and again, the maximal follow-up was the date when the data set was actually collected.

Results

The simulation demonstrated the effects of the prior on the variance and bias of the estimate of the monetization percentage. It was shown that the less information there is in the data (smaller samples or shorter follow-ups), the more the prior impacts the variance and bias of the estimate. When there is not much information in the data, the estimate is mainly based on the prior, and thus its strength affects the accuracy and precision of the estimate, as can be seen in Figure 6. For the real data sets, it was seen that in the beginning the model produces higher estimates than the current situation, but when the follow-up time is lengthened, it approaches the true value, as is shown in Figure 7 for the mobile game data set.

Author's contribution

The author was involved in the following categories of the contributor roles taxonomy [79]: data curation, formal analysis, investigation, methodology, software, visualization, writing – original draft and writing – review & editing.

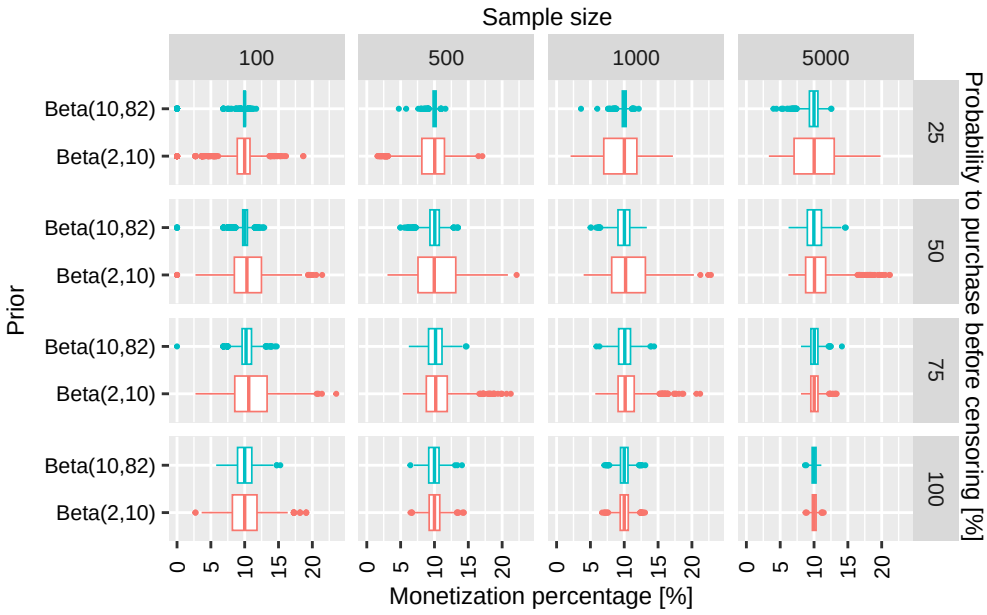

Figure 6. Boxplots of the monetization percentage estimates $\hat{\pi}$ for the given sample sizes and follow-up times (determined by probability to purchase), when the true monetization percentage $\pi = 0.10$. Figure adapted from Publication I [77], ©2020 IEEE: percentages are shown as '%' rather than the original decimal form, and figure dimensions and font are changed.

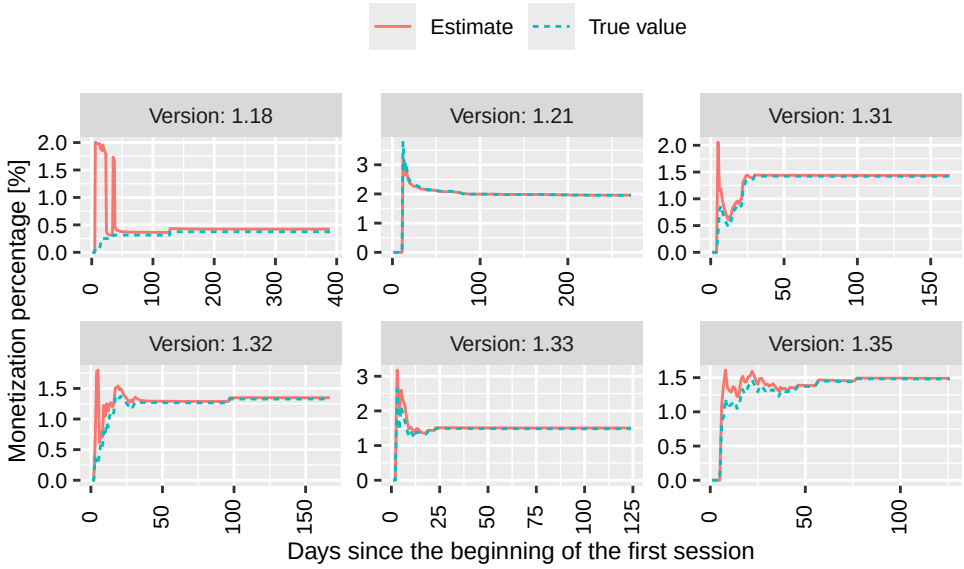

Figure 7. Monetization percentage estimates for the game data with Beta(2,50) prior. Figure adapted from Publication I [77], ©2020 IEEE: percentages are shown as '%' rather than the original decimal form, and figure dimensions and font are changed.

4.2 Quicksort Leave-Pair-Out Cross-Validation for ROC Curve Analysis

This Publication is an example of optimizing the need for resources, i.e. it is in the category of performance optimization kind of method development. The quicksort leave-pair-out (QLPO) cross-validation method presented in this work is an improved version of a previously existing method for solving the problem, as it decreases the computational complexity from $\Theta\left(n^2\right)$ to $\Theta\left(n \log n\right)$.

The problem

ROC curve analysis and AUC are often used as performance measures in medical research and diagnostic systems to measure how good a hypothesis is at distinguishing the subpopulations of having and not having a medical condition. In medical research, the sample sizes are often small, and cross-validation methods are thus used together with machine learning methods for model evaluation. Moreover, drawing an ROC curve requires determining the order of the entire data, and as a consequence, not every cross-validation method is suitable in that case. In particular, LPO is not suitable in this case, even though it is good for estimating the AUC. Pooled k-fold and LOO cross-validation methods can be used for obtaining a ranking of data, but it is created by comparing predictions made by several final hypotheses from different rounds of cross-validation. In contrast, in TLPO cross-validation, only predictions made by one hypothesis are compared, and in addition, it produces an almost unbiased estimate of AUC, as it is developed based on LPO cross-validation, but it is impractical in many applications because of its quadratic computational complexity.

The method

The presented method is developed by combining quicksort (see Section 2.3.2 for a broader description), and the idea of LPO cross-validation. Quicksort uses a divide-and-conquer paradigm for ranking the data without going through all possible pairs of data points, whereas LPO cross-validation enables the maximal use of the sample for learning a hypothesis while having the test pair separated. The ranking is obtained by selecting a pivot element $i_{pivot} \in \mathcal{I}_c$ (where $\mathcal{I}_c$ denotes the current set and is $\mathcal{I}$ in the beginning), pairing it with all the other elements within the same set $i \in \mathcal{I}_c \setminus \{i_{pivot}\}$, learning hypotheses $\hat{f}_{\mathcal{I}\setminus\{i,i_{pivot}\}}$, and then comparing the predictions $\hat{f}_{\mathcal{I}\setminus\{i,i_{pivot}\}}(i)$ and $\hat{f}_{\mathcal{I}\setminus\{i,i_{pivot}\}}(i_{pivot})$ to see whether data point i should be ranked lower or higher than the pivot. Once all the pairs in the current set are gone through and the data points are divided into subsets of smaller and greater, the same process is run recursively on the resulting subsets until a full ranking is obtained. As different hypotheses are used at every comparison step during quicksort, equally

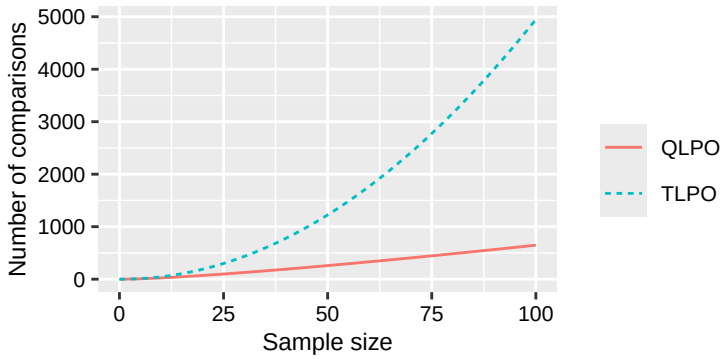

Figure 8. The effect of sample size on the number of pairs whose predicted values are compared in order to obtain a ranking by using quicksort or tournament leave-pair-out cross-validation. Figure with a small modification and caption from Publication II [80]: the label in the legend changed from QS to QLPO.

many hypotheses need to be trained as there are comparisons. On average, the number of comparisons increases more slowly as a function of sample size when QLPO cross-validation is used instead of TLPO cross-validation, as is visualized in Figure 8.

Empirical validation

The empirical validation aimed to demonstrate that the presented method produces at least equally good estimates of the ROC curve and AUC values as other suitable cross-validation methods. The quality of the method was tested by comparing its results to the results of TLPO, pooled 10-fold and LOO cross-validations, both in terms of the ROC curves and the AUC values. The experiment was run as a Monte Carlo study with 1 000 repetitions with the simulated data and 617 or 185 repetitions with the real-world data set, depending on the training sample size. Average ROC curves and AUC values were calculated over the repetitions, and variation of the estimates was measured by symmetric 95 % credible intervals. In addition, a runtime analysis was also conducted as a part of the empirical validation to give an example of the runtimes of the chosen algorithms with the four cross-validation methods.

Methods

Three learning algorithms were used to test the performance on different models. Regularised least-squares (RLS) and logistic regression (LR) are linear and parametric models, whereas random forest (RF) is a nonlinear and nonparametric ensemble method. Though the goal of the prediction task is to classify the data points, scoring algorithm versions of these methods were used to obtain the continuous val-

ues needed for evaluating the TPR and FPR with different threshold values that are needed for drawing the ROC curves.

Controlled conditions data

The simulated data were generated by a non-linear generative model with ten features $\mathbf{X} = (X_1, X_2, \ldots, X_{10})^\top$ that all followed normal distribution with mean $\mu = 0.5$ or $\mu = -0.5$ and standard deviation $\sigma = 1$. The sign of the mean value for the feature was positive with 25 % probability and negative with 75 % probability. Expression

$$\alpha \mathbf{X}^\top \beta + (1 - \alpha) \left(X_1^2 + X_2^2 + 4 X_1 X_2 \right),$$

where $\beta = (2, 1, 1, 1, 1, 0, \ldots, 0)^\top$, was scaled to range [0,1] by a sigmoid function to generate the probabilities of the datum belonging to the positive class. The coefficient α took values in the range [0,1] in steps of 0.25. Populations of a million data points were generated to correspond to the differently weighted non-linear generative models. Independent test set and 1 000 samples of 30 and 100 data points with fractions of positives of 10 % or 50 % were drawn from each population.

Real-world data

A medical data set was used to evaluate the quality of the proposed method under uncontrolled conditions. The data consisted of 85 876 voxels and six voxelwise features extracted from diffusion-weighted imaging maps [81] of 20 patients with histologically confirmed prostate cancer in the peripheral zone. The data was highly imbalanced as there were 9 268 data points with a positive label and 76 608 with a negative label. Balanced samples of sizes 30 and 100 were drawn from the data without replacement as many times as how many disjoint sets were possible to sample.

Results

Similar performance was observed both with the simulation data and real-world data: The estimates produced by QLPO and TLPO cross-validation were almost unbiased. In contrast, when pooled 10-fold or LOO cross-validation was used, the estimates were negatively biased when the sample size was 30, but almost unbiased with the larger sample size. All the credible intervals covered most of the possible area for the smaller samples and became narrower as the sample size was increased. This is illustrated in Figure 9 for the real-world data set. In other words, the prediction performance was on the same level when QLPO or TLPO cross-validation was used, and worse when either pooled 10-fold or LOO cross-validation was used.

Furthermore, the runtime analysis demonstrated the decrease in computational complexity when comparing QLPO to TLPO cross-validation, as can be seen in Ta-

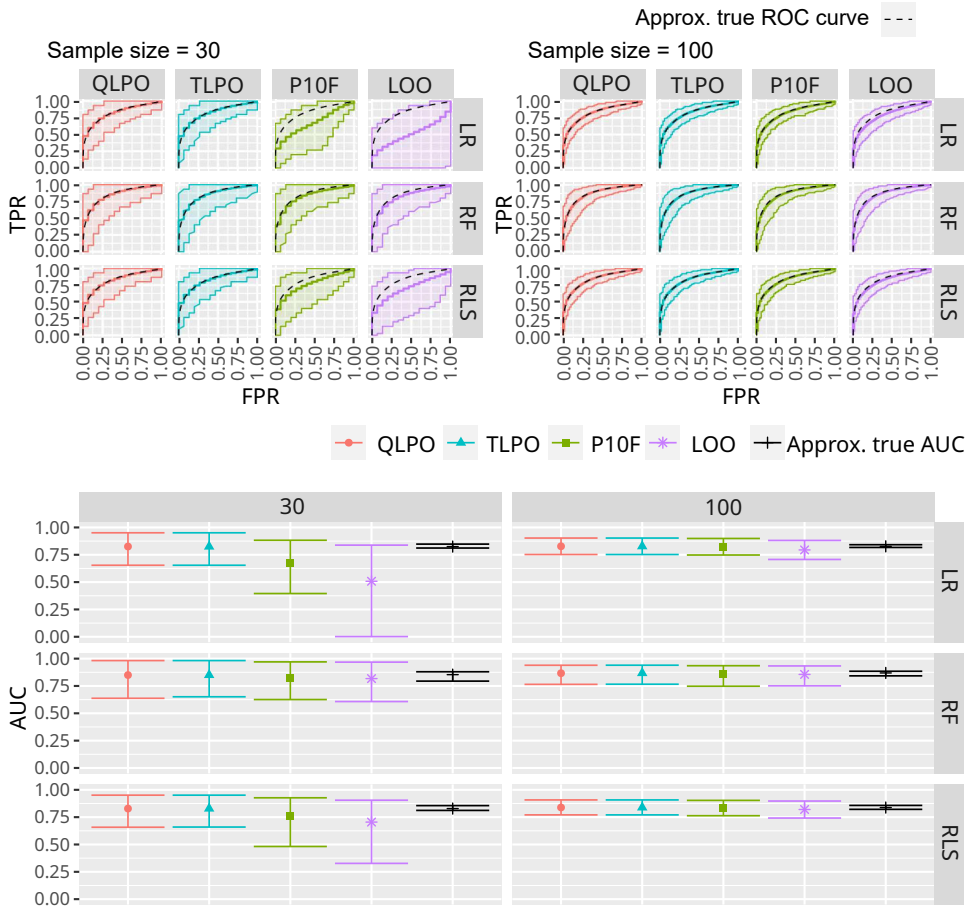

Figure 9. Average ROC curves and AUC values, and their 95 % credible intervals. Figure from Publication II [80].

ble 5. Though it was noticed that the implementation of the algorithm greatly affects the time it takes to run the different cross-validation methods: TLPO was clearly the slowest cross-validation method when the used algorithm was LR or RF, but when RLS was used, it was the fastest. This is a consequence of the use of the Sherman–Morrison–Woodbury formula in the implementation of RLS [82], because it enables faster computation of the predictions for all the pairs at once than calculating them in several smaller parts. Otherwise, the runtime analysis showed that QLPO was slower than pooled 10-fold or LOO cross-validation, as expected based on their theoretical computational complexities. Altogether, the empirical validation indicated that it is better to use QLPO cross-validation than pooled 10-fold or LOO cross-validation in order to have almost unbiased performance estimates, and QLPO is preferred to TLPO in order to have a reasonable computation time with most methods.

Table 5. Average runtimes in seconds for the cross-validation methods with RLS, LR and RF, and sample sizes 30, 50 and 100. Table adapted and caption from Publication II [80].

Sample size	30			50			100		
Algorithm	RLS	LR	RF	RLS	LR	RF	RLS	LR	RF
QLPO	0.002	0.1	20	0.005	0.3	40	0.007	0.5	70
TLPO	0.001	9	60	0.002	80	300	0.006	1500	1600
P10F	0.002	0.03	1	0.002	0.05	2	0.002	0.03	2
LOO	0.001	0.1	4	0.002	0.2	9	0.004	0.3	10

Author's contribution

The author was involved in the following categories of the contributor roles taxonomy [79]: data curation, formal analysis, investigation, methodology, software, visualization, writing – original draft and writing – review & editing.

4.3 Predicting Pairwise Interaction Affinities with ℓ_0-Penalized Least Squares – a Nonsmooth Bi-Objective Optimization Based Approach

This research study presents another example of method development in the category of generalized frameworks, where something is changed in the problem and then also in the method so that it can be used to solve the modified problem. This time, the method is a learning algorithm, and the modification is done in order to obtain sparser hypotheses that are also easier to interpret and understand, as they are less complex. The core method was kernel ridge regression, KronRLS [34], and it was generalized to enforce sparsity by using a different regularization function than usually.

The problem

Sparse hypotheses are sometimes preferred, as they are less complex and thus easier to interpret and explain. In addition, they require fewer resources when being used. In this study, the aim was to obtain these kinds of models in a large-scale pairwise interaction affinity prediction problem. In particular, in this domain, a sparse solution allows using only the most significant drug-target pairs of the training data in the prediction process. Sparsity can be obtained by using regularization to restrict the model's complexity. Depending on the regularization function, the SRM optimization problem may become nonsmooth and cannot be solved with the regularly used optimization algorithms, which are usually suitable for solving convex optimization problems.

The method

The presented learning algorithm enforces the sparsity by using an ℓ_0 pseudonorm instead of the usually used Euclidean norm in the regularization term of the SRM objective function. This change makes the objective function nonconvex:

$$\min_{\mathbf{a}} \sum_{i=1}^{n} \left(h(x_i) - y_i\right)^2 + \rho \left(||\mathbf{a}||_1 - ||\mathbf{a}||_{[k]}\right),$$

where $\mathbf{a}$ is the vector of dual coefficients in the hypothesis (1). Hence, the optimization algorithm used inside the KronRLS method is also changed from conjugate gradient to the limited memory bundle method.

In addition, main objectives, starting point procedures and bi-objective criteria are important components of the method. The main objective defines whether obtaining sparsity is a secondary goal after maintaining a high C-index (MO1) or if they are equally important goals (MO2). In the MO1, the accepted decrease in C-index is restricted to be at most a predetermined percentage of a corresponding value of the non-sparse reference model. The starting point procedure determines the initial solution when moving to the next iteration during the search for an optimal solution. It is either always the latest solution (SP1) or the best solution that was obtained with the strictest sparsity condition (SP2). Bi-objective criteria is used for model selection and is calculated as a sum of the relative C-index and sparsity:

$$cri\left(\mathbf{a}\right) = \frac{\mid \mathrm{CI}\left(\mathbf{a}\right) - \mathrm{CI}_{\text{ideal}} \mid}{\mid \mathrm{CI}_{\text{ideal}} - \mathrm{CI}_{\text{worst}} \mid} + \frac{\mid nz\left(\mathbf{a}\right) - nz_{\text{ideal}} \mid}{\mid nz_{\text{ideal}} - nz_{\text{worst}} \mid},$$

where the concordance index CI is 1 in the ideal case and 0.5 in the worst case, and the number of non-zero coefficients, nz, is 2 in the ideal case and $|\mathcal{I}_{tr}|$ in the worst case. This kind of bi-objective criterion enables the relative changes in the generalization performance and sparsity to affect the criterion equally.

Empirical validation

The quality of the presented method was experimentally tested by comparing its performance to the performances of baseline methods on two simulated data sets and three biomedical interaction data sets. Both continuous and binarized labels were separately used. In addition, the performance was tested in the four different learning settings depending on whether the drugs and targets were in- or off-training-set drugs and targets.

Methods

The presented method was compared to KronRLS and KronSVM, which are state-of-the-art learning algorithms for predicting the pairwise interactions. KronSVM can produce sparse models, so in the comparison with this algorithm, the performance was evaluated both in terms of sparsity as well as in the generalization ability. Sparsity was measured by the percentage of coefficients equal to zero, and generalization ability by the C-index. KronRLS, on the other hand, does not produce sparse models, but its benefit is that it can be used both with continuous and binary labels, unlike KronSVM, which is limited to the classification task only.

Controlled conditions data

The simulated data sets were sparse variants of the checkerboard simulation with and without added noise. Each drug and target had a descriptive feature following a continuous uniform distribution $U(0, 5)$. The labels were determined by the XOR function on the even parities of the features describing the drug and the target in the binary case. Continuous labels were produced by multiplying the binary label by both feature values associated with the pair. Separate training, validation and test sets were generated according to the described procedure with different random seeds. In the training and validation sets, there were 100 drugs and 100 targets. Test set, then again, contained 500 drugs and 500 targets. All sets were sparse with labels assigned to randomly selected 25 % of the pairs. Noise was added by flipping the sign of the label of each pair with 20 % probability. These data sets were generated for having such data that the methods run fast on them.

Real-world data

Three benchmark data sets were used for testing the method on real-world data. Metz [83] and Merget [84] data sets were sparse, whereas the Davis [85] had all the drug-target interaction affinities known. Related to Davis and Metz data sets, the drug feature matrices contained structural fingerprint similarities between two drugs

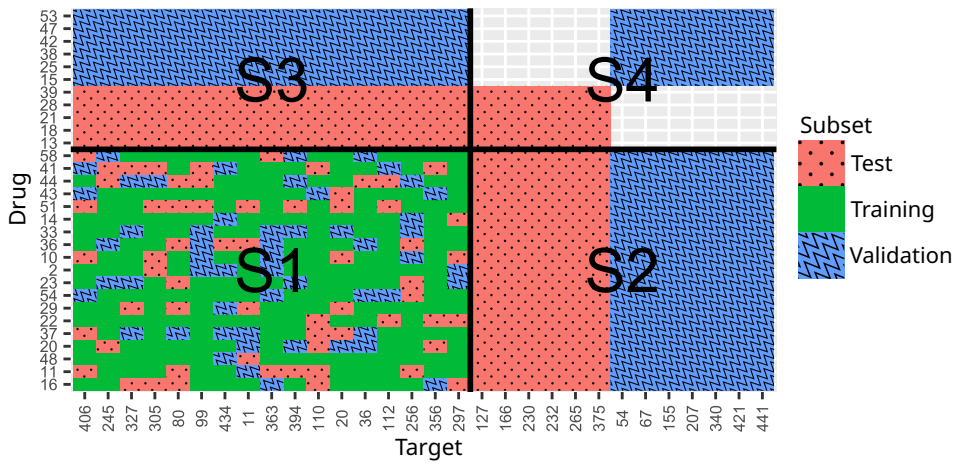

Figure 10. A generated example for visually demonstrating the used data splits. S1 refers to a generalization task, where both the drugs and the targets of test and validation pairs also exist in the training set. S2 refers to a generalization task, where the test and validation targets do not exist in the training set, but the test and validation drugs do. In contrast, S3 refers to a generalization task, where the test and validation drugs do not exist in the training set, but the test and validation targets do. Finally, S4 refers to the cold-start problem, where neither the drugs nor the targets of test and validation pairs exist in the training set. Figure from Publication III [42] with a larger font size in the legend.

computed with 2D Tanimoto coefficients and the target feature matrices contained the normalized version of the Smith-Waterman scores between two targets. The interaction affinities were quantified as the dissociation constant in Davis data and the inhibition constant in Metz data. For the Merget data, the structural fingerprints were computed by 1024-bit fingerprint based on the shortest path between atoms, the target feature matrix contained amino acid sub-sequences of ATP binding pockets and amino acid properties, and the interactions were measured by binding affinities. As the labels were in different units, the binarization thresholds were also different for the data sets: 30 for Davis, 7.6 for Metz, and 7 for Merget. The biomedical data sets were split into training, validation and test sets in a way that the training set was the same in all generalization tasks, and validation and test sets were selected so that they fulfilled the conditions for the four different generalization tasks, which were described in Section 2.1.2. An example of this kind of data split is given in Figure 10.

Results

The results were reported as sparsity percentage and C-index values. Using the presented method with MO1 resulted in a clearly sparser solution than KronRLS without sacrificing too much in terms of the prediction accuracy. Using the MO2 led to even

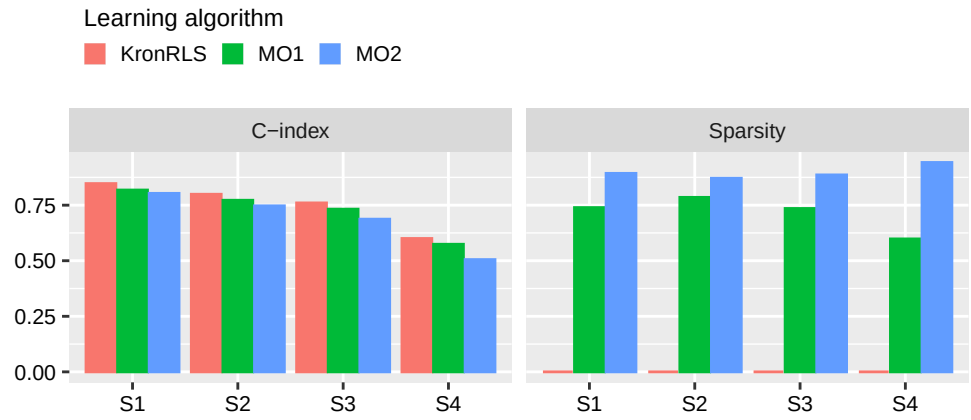

Figure 11. A visualization of the results for the Davis data set with continuous labels. S1 refers to a generalization task, where both the drugs and the targets of test and validation pairs also existed in the training set. S2 refers to a generalization task, where the test and validation targets do not exist in the training set, but the test and validation drugs do. In contrast, S3 refers to a generalization task, where the test and validation drugs do not exist in the training set, but the test and validation targets do. Finally, S4 refers to the cold-start problem, where neither the drugs nor the targets of test and validation pairs exist in the training set.

sparser solutions, as the decrease in C-index was not restricted. Even the comparison to KronSVM, in the binary case, showed that the presented methods found notably sparser solutions than the state-of-the-art methods without decreasing the C-index too much. An example of the results is given in Figure 11. It can be seen that the generalization performance remains competitive in most cases, while the sparsity increases significantly in all cases.

Author's contribution

The author was involved in the following categories of the contributor roles taxonomy [79] during a revision phase: data curation, formal analysis, investigation (re-running the experiment with the real-world data set and adding two other real-world data sets), software, visualization and writing – review & editing.

5 Conclusion and Future Work

The main objective of this thesis was to unite three research studies, where varying types of data science method development had been done. The problems in the studies were different, and thus also the developed methods are related to different areas of data science – though, in all of the studies, the goal was to predict something, which limited the scope of this thesis to predictive data science. One of the studies was related to the generative data science approach, whereas the other two used the discriminative approach. The presented methods were about formulating a survival analysis mixture cure model in the field of free-to-play mobile games, developing a faster cross-validation method (QLPOCV) that can be used together with ROC curve analysis, and modifying a learning algorithm for pairwise data to enforce sparsity and thus produce simpler hypotheses without compromising the generalization ability considerably. Thus, the method development category was generalized frameworks in the first and third studies, and performance optimization in the second study.

Even though the method development process was presented to contain both theoretical analysis and empirical validation steps, the methods presented in the research studies were evaluated mainly empirically. In all of the studies, the empirical validation consisted of using both simulated controlled conditions data and real-world data. The controlled conditions data were used for estimating the effects of different assumptions of the data or of a component of the method on the performance of the method: e.g. the effect of the choice of the prior distribution in the Bayesian inference, the impact of linearity of the data generative model on the ability to rank the data correctly, and controlled amount of noise in the data when learning sparse hypotheses. The real-world data sets, then again, were used to see performances of the methods in uncontrolled conditions. However, the empirical validation focused on one domain per research study, and thus, it is not evaluated how the methods would generalize to different domains. Nevertheless, the methods are not domain-specific and could be applied to similar tasks in other domains, as well, while keeping in mind that the monetization model is developed for time-to-event data and the LMBMKronℓ_0LS for pairwise data.

In method development, there is always plenty of future work that can be done. In particular, several arose questions remained unresolved in the research studies: In the monetization prediction task, the partial violation of the required parametric assumptions had a greater impact on the results than expected, and thus reassessing

the theoretical assumptions could improve the method. Secondly, when it comes to QLPOCV, theoretical research remains to be done to estimate the effect of possible cycles in the corresponding tournament graph, whose existence causes that the order of data points may not be unequivocal but instead depend on the random selection of the pivot elements. On the other hand, regarding the LMBMKronℓ_0LS method, further empirical analysis should also be conducted in order to evaluate possible random effects that may have appeared because of the used random split of data, as only a single split was used instead of some kind of resampling. In addition, the LMBMKronℓ_0LS method is time demanding and rather complicated to use. The usability of the methods is a common problem, as the focus has been on describing the ideas of the methods and how they were used, and wrapping them in a form that would be easy to use remains to be done.

List of References

[1] Longbing Cao. Data science: A comprehensive overview. *ACM Computing Surveys*, 50(3):1–42, 2017. doi: ⟨https://doi.org/10.1145/3076253⟩.

[2] Mark Baillie, Conor Moloney, Carsten P. Mueller, Jonas Dorn, Janice Branson, and David Ohlssen. Good data science practice: Moving toward a code of practice for drug development. *Statistics in Biopharmaceutical Research*, 15(1):74–85, 2023. doi: ⟨https://doi.org/10.1080/19466315.2022.2063172⟩.

[3] David M. Blei and Padhraic Smyth. Science and data science. *Proceedings of the National Academy of Sciences*, 114(33):8689–8692, 2017. doi: ⟨https://doi.org/10.1073/pnas.1702076114⟩.

[4] Christian Heumann, Michael Schomaker, and Shalabh. *Introduction to Statistics and Data Analysis: With Exercises, Solutions and Applications in R*. Springer International Publishing Switzerland, 2016. doi: ⟨https://doi.org/10.1007/978-3-319-46162-5⟩.

[5] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag New York, 2nd edition, 2009. doi: ⟨https:doi.org/10.1007/978-0-387-84858-7⟩.

[6] Ralph L. Keeney. Decision analysis: An overview. *Operations Research*, 30(5):803–838, 1982. doi: ⟨https://doi.org/10.1287/opre.30.5.803⟩.

[7] Howard M. Taylor and Samuel Karlin. *An Introduction to Stochastic Modeling*. Academic Press, 3rd edition, 1998.

[8] Leslie P. Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. doi: ⟨https://doi.org/10.1613/jair.301⟩.

[9] Leo Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–231, 2001. doi: ⟨https://doi.org/10.1214/ss/1009213726⟩.

[10] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. doi: ⟨https://doi.org/10.1017/CBO9781107298019⟩.

[11] Constantin Aliferis and Gyorgy Simon. Principles of rigorous development and of appraisal of ML and AI methods and systems. In Constantin Aliferis and Simon Gyorgy, editors, *Artificial Intelligence and Machine Learning in Health Care and Medical Sciences: Best Practices and Pitfalls*, pages 229–288. Springer Nature Switzerland AG, 2024. doi: ⟨https://doi.org/10.1007/978-3-031-39355-6_5⟩.

[12] Ralph Deutsch. *Estimation Theory*. Prentice-Hall, Inc., 1965.

[13] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[14] David R. Cox and David Oakes. *Analysis of Survival Data*. Chapman and Hall Ltd, 1984.

[15] Markus Viljanen, Antti Airola, Jukka Heikkonen, and Tapio Pahikkala. Playtime measurement with survival analysis. *IEEE Transactions on Games*, 10(2):128–138, 2017. doi: ⟨https://doi.org/10.1109/TCIAIG.2017.2727642⟩.

[16] Yingwei Peng and Jeremy M. G. Taylor. Cure models. In John P. Klein, Hans C. van Houwelingen, Joseph G. Ibrahim, and Thomas H. Scheike, editors, *Handbook of Survival Analysis*, volume 34, pages 113–134. Chapman and Hall/CRC, 2014. doi: ⟨https://doi.org/10.1201/b16248⟩.

[17] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. doi: ⟨https://doi.org/10.1111/j.2517-6161.1977.tb01600.x⟩.

[18] Markus Viljanen, Antti Airola, and Tapio Pahikkala. Generalized vec trick for fast learning of pairwise kernel models. *Machine Learning*, 111:543–573, 2022. doi: ⟨https://doi.org/10.1007/s10994-021-06127-y⟩.

[19] Tapio Pahikkala, Antti Airola, Sami Pietilä, Sushil Shakyawar, Agnieszka Szwajda, Jing Tang, and Tero Aittokallio. Toward more realistic drug–target interaction predictions. *Briefings in Bioinformatics*, 16(2):325–337, 2015. doi: ⟨https://doi.org/10.1093/bib/bbu010⟩.

[20] Jason W. Rocks and Pankaj Mehta. Memorizing without overfitting: Bias, variance, and interpolation in overparameterized models. *Physical Review Research*, 4:013201, 2022. doi: ⟨https://doi.org/10.1103/PhysRevResearch.4.013201⟩.

[21] Thomas Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003. doi: ⟨https://doi.org/10.1145/959242.959248⟩.

[22] Alain Berlinet and Christine Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer Science + Business Media, LLC, 2004. doi: ⟨https://doi.org/10.1007/978-1-4419-9096-9⟩.

[23] Timothy Sauer. *Numerical Analysis*. Pearson Eduation, Inc., 2nd edition, 2012.

[24] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC Texts in Statistical Science. Chapman and Hall/CRC, 3rd edition, 2013. doi: ⟨https://doi.org/10.1201/b16018⟩.

[25] Balas K. Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995. doi: ⟨https://doi.org/10.1137/S0097539792240406⟩.

[26] Henrik Ohlsson. *Regularization for Sparseness and Smoothness Applications in System Identification and Signal Processing*. PhD thesis, Linköping University, 2010.

[27] Manlio Gaudioso, Enrico Gorgone, and J.-B. Hiriart-Urruty. Feature selection in SVM via polyhedral k-norm. *Optimization Letters*, 14:19–36, 2020. doi: ⟨https://doi.org/10.1007/s11590-019-01482-1⟩.

[28] Jun-ya Gotoh, Akiko Takeda, and Katsuya Tono. DC formulations and algorithms for sparse optimization problems. *Mathematical Programming, Series B*, 169:141–176, 2018. doi: ⟨https://doi.org/10.1007%2Fs10107-017-1181-0⟩.

[29] Mokhtar S. Bazaraa, Hanif D. Sherali, and Chitharanjan M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John wiley & sons, Inc., 3rd edition, 2006. doi: ⟨https://doi.org/10.1002/0471787779⟩.

[30] Marc P. Deisenroth, A. Aldo Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning*. Cambridge University Press, 2020. doi: ⟨https://doi.org/10.1017/9781108679930⟩.

[31] Adil M. Bagirov, Manlio Gaudioso, Napsu Karmitsa, Marko M. Mäkelä, and Sona Taheri. *Numerical Nonsmooth Optimization: State of the Art Algorithms*. Springer Nature Switzerland AG, 2020. doi: ⟨https://doi.org/10.1007/978-3-030-34910-3⟩.

[32] Napsu Karmitsa. Limited memory bundle method and its variations for large-scale nonsmooth optimization. In Adil M. Bagirov, Manlio Gaudioso, Napsu Karmitsa, Marko M. Mäkelä, and Sona Taheri, editors, *Numerical Nonsmooth Optimization: State of the Art Algorithms*, pages 167–199. Springer Nature Switzerland AG, 2020. doi: ⟨https://doi.org/10.1007/978-3-030-34910-3_5⟩.

[33] Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. John Wiley & Sons, Inc., 1st edition, 1997.

[34] Tapio Pahikkala and Antti Airola. RLScore: Regularized least-squares learners. *Journal of Machine Learning Research*, 17(220):1–5, 2016. URL <http://jmlr.org/papers/v17/16-470.html>.

[35] Ryan M. Rifkin and Ross A. Lippert. Notes on regularized least squares. Technical report, MIT, 2007. URL <http://hdl.handle.net/1721.1/37318>.

[36] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Oliver Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL <https://dl.acm.org/doi/10.5555/1953048.2078195>.

[37] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008. URL <http://jmlr.org/papers/v9/fan08a.html>.

[38] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. doi: ⟨https://doi.org/10.1023/A:1010933404324⟩.

[39] Leo Breiman, Jerome Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984. doi: ⟨https://doi.org/10.1201/9781315139470⟩.

[40] Tapio Pahikkala. Fast gradient computation for learning with tensor product kernels and sparse training labels. In Pasi Fränti, Gavin Brown, Marco Loog, Francisco Escolano, and Marcello Pelillo, editors, *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 123–132. Springer Berlin Heidelberg, 2014. doi: ⟨https://doi.org/10.1007/978-3-662-44415-3_13⟩.

[41] Roland W. Freund and Noël M. Nachtigal. QMR: A quasi-minimal residual method for non-Hermitian linear systems. *Numerische Mathematik*, 60(1):315–339, 1991. doi: ⟨https://doi.org/10.1007/BF01385726⟩.

[42] Pauliina Paasivirta, Riikka Numminen, Antti Airola, Napsu Karmitsa, and Tapio Pahikkala. Predicting pairwise interaction affinities with ℓ_0-penalized least squares–a nonsmooth bi-objective optimization based approach. *Optimization Methods and Software*, 41(2):450–477, 2026. doi: ⟨https://doi.org/10.1080/10556788.2023.2280784⟩.

[43] Antti Airola and Tapio Pahikkala. Fast Kronecker product kernel methods via generalized vec trick. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8):3374–3387, 2018. doi: ⟨https://doi.org/10.1109/TNNLS.2017.2727545⟩.

[44] Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011. doi: ⟨https://doi.org/10.1017/CBO9780511921803⟩.

[45] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. doi: ⟨https://doi.org/10.1109/TAC.1974.1100705⟩.

[46] David E. Shapiro. The interpretation of diagnostic tests. *Statistical Methods in Medical Research*, 8(2):113–134, 1999. doi: ⟨https://doi.org/10.1177/096228029900800203⟩.

[47] James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982. doi: ⟨https://doi.org/10.1148/radiology.143.1.7063747⟩.

[48] Nathalie Japkowicz and Mohak Shah. Performance evaluation in machine learning. In Issam El Naqa, Ruijiang Li, and Martin J. Murphy, editors, *Machine Learning in Radiation Oncology: Theory and Applications*, pages 41–56. Springer International Publishing Switzerland, 2015. doi: ⟨https://doi.org/10.1007/978-3-319-18305-3_4⟩.

[49] Andrew P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997. doi: ⟨https://doi.org/10.1016/S0031-3203(96)00142-2⟩.

[50] Antti Airola, Tapio Pahikkala, Willem Waegeman, Bernard De Baets, and Tapio Salakoski. An experimental comparison of cross-validation techniques for estimating the area under the ROC curve. *Computational Statistics & Data Analysis*, 55(4):1828–1844, 2011. doi: ⟨https://doi.org/10.1016/j.csda.2010.11.018⟩.

[51] Ileana Montoya Perez, Antti Airola, Peter J. Boström, Ivan Jambor, and Tapio Pahikkala. Tournament leave-pair-out cross-validation for receiver operating characteristic analysis. *Statistical Methods in Medical Research*, 28(10-11):2975–2991, 2019. doi: ⟨https://doi.org/10.1177/0962280218795190⟩.

[52] Michael A. Lones. Avoiding common machine learning pitfalls. *Patterns*, 5(10), 2024. doi: ⟨https://doi.org/10.1016/j.patter.2024.101046⟩.

[53] Colin Shearer. The CRISP-DM model: The new blueprint for data mining. *Journal of Data Warehousing*, 5(4):13–22, 2000.

[54] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996. doi: ⟨https://doi.org/10.1145/240455.240464⟩.

[55] SAS Enterprise Miner. Introduction to SEMMA. SAS Institute. URL <https://documentation.sas.com/doc/en/emref/14.3/n061bzurmej4j3n1jnj8bbjjm1a2.htm>. Accessed: 11 June 2025.

[56] Microsoft. Machine learning operations. URL <https://learn.microsoft.com/en-us/azure/architecture/ai-ml/guide/machine-learning-operations-v2>. Accessed: 11 June 2025.

[57] Hilary Mason and Chris Wiggins. A taxonomy of data science. Data Science @ ISiM, 2010. URL <https://introdatasci.dlilab.com/pdf/A_Taxonomy_of_Data_Science.pdf>.

[58] Iñigo Martinez, Elisabeth Viles, and Igor G. Olaizola. Data science methodologies: Current challenges and future approaches. *Big Data Research*, 24:100183, 2021. doi: ⟨https://doi.org/10.1016%2Fj.bdr.2020.100183⟩.

[59] Sallie A. Keller, Stephanie S. Shipp, Aaron D. Schroeder, and Gizem Korkmaz. Doing data science: A framework and case study. *Harvard Data Science Review*, 2(1), 2020. doi: ⟨https://doi.org/10.1162/99608f92.2d83f7f5⟩.

[60] Jeannette M. Wing. The data life cycle. *Harvard Data Science Review*, 1(1):6, 2019. doi: ⟨https://doi.org/10.1162/99608f92.e26845b4⟩.

[61] Victoria Stodden. The data science life cycle: A disciplined approach to advancing data science as a science. *Communications of the ACM*, 63(7):58–66, 2020. doi: ⟨http://dx.doi.org/10.1145/3360646⟩.

[62] Sayash Kapoor, Emily M. Cantrell, Kenny Peng, Thanh Hien Pham, Christopher A. Bail, Odd Erik Gundersen, Jake M. Hofman, Jessica Hullman, Michael A. Lones, Momin M. Malik, et al. REFORMS: Consensus-based recommendations for machine-learning-based science. *Science Advances*, 10(18):eadk3452, 2024. doi: ⟨https://doi.org/10.1126/sciadv.adk3452⟩.

[63] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (A report from the NeurIPS 2019 Reproducibility Program). *Journal of Machine Learning Research*, 22(164):1–20, 2021. URL <http://jmlr.org/papers/v22/20-303.html>.

[64] Ayomide Owoyemi, Joanne Osuchukwu, Megan E. Salwei, and Andrew Boyd. Checklist approach to developing and implementing AI in clinical settings: Instrument development study. *JMIRx Med*, 6:e65565, 2025. doi: ⟨https://doi.org/10.2196/65565⟩.

[65] Gary S. Collins, Johannes B. Reitsma, Douglas G. Altman, and Karel G. M. Moons. Transparent reporting of a multivariable prediction model for individual prognosis or diagnosis: The TRIPOD statement. *Circulation*, 131(2):211–219, 2015. doi: ⟨https://doi.org/10.1161%2FCIRCULATIONAHA.114.014508⟩.

[66] John Mongan, Linda Moy, and Charles E. Kahn Jr. Checklist for artificial intelligence in medical imaging (CLAIM): A guide for authors and reviewers. *Radiology: Artificial Intelligence*, 2(2):e200029, 2020. doi: ⟨https://doi.org/10.1148/ryai.2020200029⟩.

[67] Sang Gyu Kwak and Jonghae Kim. Comprehensive reporting guidelines and checklist for studies developing and utilizing artificial intelligence models. *Korean Journal of Anesthesiology*, 78(3):199–214, 2025. doi: ⟨https://doi.org/10.4097/kja.25075⟩.

[68] Seung-Ho Han and Ho-Jin Choi. Checklist for validating trustworthy AI. In *2022 IEEE International Conference on Big Data and Smart Computing*, pages 391–394. IEEE, 2022. doi: ⟨https://doi.org/10.1109%2FBigComp54360.2022.00088⟩.

[69] Salah S. Al-Zaiti, Alaa A. Alghwiri, Xiao Hu, Gilles Clermont, Aaron Peace, Peter Macfarlane, and Raymond Bond. A clinician's guide to understanding and critically appraising ma-

chine learning studies: A checklist for ruling out bias using standard tools in machine learning (ROBUST-ML). *European Heart Journal - Digital Health*, 3(2):125–140, 2022. doi: ⟨https://doi.org/10.1093/ehjdh/ztac016⟩.

[70] Nabeel Seedat, Fergus Imrie, and Mihaela van der Schaar. DC-check: A data-centric AI checklist to guide the development of reliable machine learning systems. *arXiv preprint arXiv:2211.05764*, 2022. doi: ⟨https://doi.org/10.1109/TAI.2023.3345805⟩.

[71] Partho P. Sengupta, Sirish Shrestha, Béatrice Berthon, Emmanuel Messas, Erwan Donal, Geoffrey H. Tison, James K. Min, Jan D'hooge, Jens-Uwe Voigt, Joel Dudley, et al. Proposed requirements for cardiovascular imaging-related machine learning evaluation (PRIME): A checklist: Reviewed by the American College of Cardiology Healthcare Innovation Council. *Cardiovascular Imaging*, 13(9):2017–2035, 2020. doi: ⟨https://doi.org/10.1016/j.jcmg.2020.07.015⟩.

[72] Zsombor Zrubka, László Gulácsi, and Márta Péntek. Time to start using checklists for reporting artificial intelligence in health care and biomedical research: A rapid review of available tools. In *2022 IEEE 26th International Conference on Intelligent Engineering Systems*, pages 000015–000020. IEEE, 2022. doi: ⟨https://doi.org/10.1109/INES56734.2022.9922639⟩.

[73] A. Stevie Bergman, Lisa A. Hendricks, Maribeth Rauh, Boxi Wu, William Agnew, Markus Kunesch, Isabella Duan, Iason Gabriel, and William Isaac. Representation in AI evaluations. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 519–533, New York, NY, USA, 2023. Association for Computing Machinery. doi: ⟨https://doi.org/10.1145/3593013.3594019⟩.

[74] Christoph Molnar. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Online book, 2020. URL <https://christophm.github.io/interpretable-ml-book/>. Accessed: 2026-04-23.

[75] Phillip Good. *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses*. Springer Series in Statistics. Springer New York, NY, 2000. doi: ⟨https://doi.org/10.1007/978-1-4757-2346-5⟩.

[76] David J. Hand. Classifier technology and the illusion of progress. *Statistical Science*, 21(1):1–14, 2006. doi: ⟨https://doi.org/10.1214%2F088342306000000060⟩.

[77] Riikka Numminen, Markus Viljanen, and Tapio Pahikkala. Bayesian inference for predicting the monetization percentage in free-to-play games. *IEEE Transactions on Games*, 14(1):13–22, 2022. doi: ⟨https://doi.org/10.1109/TG.2020.3014660⟩.

[78] Peter S. Fader and Bruce G. S. Hardie. Forecasting repeat sales at CDNOW: A case study. *Interfaces*, 31(3_supplement):S94–S107, 2001. doi: ⟨https://doi.org/10.1287/inte.31.3s.94.9683⟩.

[79] Amy Brand, Liz Allen, Micah Altman, Marjorie Hlava, and Jo Scott. Beyond authorship: Attribution, contribution, collaboration, and credit. *Learned Publishing*, 28(2):151–155, 2015. doi: ⟨https://doi.org/10.1087/20150211⟩.

[80] Riikka Numminen, Ileana Montoya Perez, Ivan Jambor, Tapio Pahikkala, and Antti Airola. Quicksort leave-pair-out cross-validation for ROC curve analysis. *Computational Statistics*, 38:1579–1595, 2023. doi: ⟨https://doi.org/10.1007/s00180-022-01288-3⟩.

[81] Jussi Toivonen, Harri Merisaari, Marko Pesola, Pekka Taimen, Peter J. Boström, Tapio Pahikkala, Hannu J. Aronen, and Ivan Jambor. Mathematical models for diffusion-weighted imaging of prostate cancer using b values up to 2000 s/mm2: Correlation with Gleason score and repeatability of region of interest analysis. *Magnetic Resonance in Medicine*, 74(4):1116–1124, 2015. doi: ⟨https://doi.org/10.1002/mrm.25482⟩.

[82] Tapio Pahikkala, Antti Airola, Jorma Boberg, and Tapio Salakoski. Exact and efficient leave-pair-out cross-validation for ranking RLS. In *Proceedings of the 2nd International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 1–8, Espoo, Finland, 2008. Helsinki University of Technology.

[83] James T. Metz, Eric F. Johnson, Niru B. Soni, Philip J. Merta, Lemma Kifle, and Philip J. Hadjuk. Navigating the kinome. *Nature Chemical Biology*, 7:200–202, 2011. doi: ⟨https://doi.org/10.1038/nchembio.530⟩.

[84] Benjamin Merget, Samo Turk, Sameh Eid, Friedrich Rippmann, and Simone Fulle. Profiling prediction of kinase inhibitors: Toward the virtual assay. *Journal of Medicinal Chemistry*, 60(1): 474–485, 2017. doi: ⟨http://dx.doi.org/10.1021/acs.jmedchem.6b01611⟩.

[85] Mindy I. Davis, Jeremy P. Hunt, Sanna Herrgard, Pietro Ciceri, Lisa M. Wodicka, Gabriel Pallares, Michael Hocker, Daniel K. Treiber, and Patrick P. Zarrinkar. Comprehensive analysis of kinase inhibitor selectivity. *Nature Biotechnology*, 29(11):1046–1051, 2011. doi: ⟨https://doi.org/10.1038/nbt.1990⟩.

TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

Painosalama, Turku, Finland 2026

ISBN 978-952-02-0775-5 (PRINT)
ISBN 978-952-02-0776-2 (PDF)
ISSN 2736-9390 (PRINT)
ISSN 2736-9684 (ONLINE)