JOHANNES KEPLER
UNIVERSITY LINZ

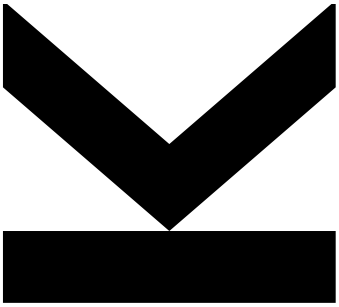
Submitted by
Georg Hermanutz

Submitted at
Institute of Bioinformatics

Supervisor
Assoz.Univ.-Prof. Dipl.-Ing. Dr. Ulrich Bodenhofer

July 2017

Software using random forest for risk prediction of heart valve surgery patients.



Bachelor Thesis
to obtain the academic degree of
Bachelor of Science
in the Bachelor's Program
Bioinformatics

JOHANNES KEPLER
UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Abstract

CASPeR - Cardiac surgery prediction tool, an application that facilitates risk stratification for heart valve surgeries is presented. CASPeR utilizes a machine learning pipeline that trains a random forest classifier to predict the mortality after a certain amount of days after the surgery. The training is done with the R package 'randomForest'. Prediction is done using out-of-bag samples only. Based on the prediction, a survival analysis is carried out and visualized with a Kaplan-Meier plot. A log-rank test is used to test for statistical significance. The R package 'survival' is utilized to carry out the survival analysis as well as the log-rank test. To identify potential risk factors, random forests feature selection is used, which is displayed in a variable importance plot. Instead of using the mean decrease in accuracy and the mean decrease in Gini impurity to rank the importance of a variable for the model, it was deduced that the mean decrease in Gini impurity holds more information. The R package Shiny, which offers a web framework to develop data analysis visualizations is used to build a user interface that guides doctors through each step of the pipeline. CASPeR is designed to be a local desktop application that can be run offline, to avoid the necessity of large dataset uploads as well as to offer online usability. It comes with a Microsoft Windows compatible installer and a detailed Manual. Additionally, CASPeR was tested on a large dataset containing data obtained from 2,229 cardiac valve surgery patients. The data contained features such as age, weight, height, ASA, parameters obtained from A-IQI Austrian Inpatient Quality Indicators, and preoperative laboratory parameters. 37% of the data were missing values, which were taken care of by a sophisticated preprocessing step implementated in CASPeR. The initial mortality determined from the data was 3.848%. The random forest obtained with CASPeR achieved an AUC of 0.8399, optimized for accuracy, and 0.804 when optimized for balanced accuracy. The mortality for the high-risk group, i.e. patients predicted as dead after 30 days after the surgery, was found to increase from 3.858% (observed) to 13.89% (predicted). CASPeR can be downloaded from `www.dropbox.com/sh/9b47s66d0mjny1`.

Contents

Acknowledgement

First and foremost, I have to thank my research supervisor, Prof. Bodenhofer. Without his patient guidance, enthusiastic encouragement and useful critique, this work would have never been accomplished. He was also my 'Machine Learning: Supervised Techniques' professor at Johannes Kepler University. His teaching style and enthusiasm for the topic made a strong impression on me and ultimately led me into the world of machine learning. Thank you very much for your support over the the past years.

I would like to express my deep gratitude Prof. Meier and Mag. Haslinger-Eisterer for being genuinely nice during our meetings, even when my face became pale due to my nervousness. I also want to thank them for providing me with valuable information about medical data and contributing ideas to the design of the User Interface. I must also thank Dr. med. univ. Povysil, who helped me out with her experience with Shiny.

Getting through the debugging required more than academic support, and I have many, many people to thank for listing to and, at times having to tolerate me over the past years. I cannot express my gratitude and appreciation for their friendship. Costadedoi, Ringhofer, Baumgartner were responsible for many memorable nights out.

Most importantly, none of this could have happened without my family. My grandmothers, who offered their encouragement, even during periods of bad health conditions. To my parents and my brother - it would be an understatement to say that, as a family, we experienced some ups and downs in the past four years. I want to deeply thank your for the unceasing encouragement, support and attention, even during chaotic times. This thesis stands as a testament to your unconditional love and encouragement.

Chapter 1

Introduction

Over the course of nearly 100 years, heart valve surgery developed from being unknown to a highly standardized procedure, with comparatively low mortality rates. Although, many different risk factors are known and well researched, assessing the risk profile of an individual heart surgery patient remains a highly difficult task. Nowadays different scoring systems are used to allow doctors the determination of the risk profile. These systems, usually are based on regression models, which were trained on risk factors that were found to be indicative for predicting the outcome.

One such system and the most prominent one is the EuroSCORE - European System for Cardiac Operative Risk Evaluation [18, 19]. The original model EuroSCORE I is based on 17 variables, while the updated version EuroSCORE II is based on 19 variables. These variables were all identified by a regression model [26]. However, such models might misrepresent the importance of other variables or even more so neglect the importance that bears with the influence of one variable on another in a given circumstance. The latter is completely neglected in both EuroSCORE systems because the risk score is calculated by adding weight to the 17/19 factors and sum them up. The resulting score yields the approximate percent predicted mortality. Moreover, it has been shown that systems like EuroSCORE I and II cannot be used to predict the risk of an individual patient and are limited when used on certain subgroups which were not taken into account during the development of the system. A field that gained a lot of traction over the past couple of years, and could solve the drawbacks traditional risk assessment systems have, is machine learning. A technique that has shown good results in a recent study [34], is the random forest technique (short RF) [6]. It could overcome the drawbacks of the traditional system, since it does not work with the assumption of a linear relationship but rather work in a nonlinear way, which might be able to represent the relationship between risk factors better. Currently work is done to investigate if machine learning algorithms indeed perform better in risk

stratification than traditional systems.

RF consists of an ensemble of decision trees and is nonparametric, i.e the data points themselves are the parameters which make up the model. It naturally incorporates feature selection in the learning process and has a high prediction accuracy.Additionally, RF can deal with highly imbalanced class distributions such as the observed mortality rate of heart valve surgeries (3.992% [17]). Unfortunately, machine learning methods require a high level of expertise to produce meaningful results. Usually, clinicians do not reach this level of expertise and would not be able to apply the potentially superior system to their data. Although, through the recent release of the R package Shiny [7] by Rstudio, it is now possible for developers to create application that allow the use of data analysis pipelines even without knowledge about R. The aim of this work was to provide a Shiny application that would allow a user to use machine learning packages, such as randomForest [14], ROCR,..., to train a random forest classifier. Furthermore, this classifier would be used in combination with a Kaplan-Meier survival analysis to investigate the mortality an arbitrary number of days after the surgery. The natural feature selection, the randomForest technique provides, offers a perfect tool to assess the most important risk factors for the model. The application should be distributed as a local standalone application, which brought the challenge of wrapping the R environment and all the necessary packages into a Microsoft Windows compatible installer.

This report will give an introduction to machine learning and the random forest technique, i.e from decision trees to random forests and the attributes that make this technique so viable. In addition, an explanation of how machine learning models are evaluated as well as how a Kaplan-Meier analysis works is given. The second half of the report consists of the architecture and implementation details. Moreover, distribution aspects of a Shiny application are discussed. Finally, a short description of the analysis workflow, as well as a description of the user interface elements, is given. To round everything up, a case study will be discussed.

Chapter 2

Machine Learning

In the past years, machine learning became an increasingly important field /tool for many different applications, be it speech and handwriting recognition, online advertisements, bioinformatics, medicine or one of many more. This manifold applicability of machine learning comes from its ability to construct algorithms that learn and make predictions from data. Machine learning evolved from the study of pattern recognition and computational learning theory in artificial intelligence. It is also closely related to computational statistics and borrows many methods, theories and application domains from mathematical optimization. While machine learning has a lot of similarities with data mining, it is more about predictive analyses and not so much about exploratory data analysis.

Machine learning consists of 3 different subfields: supervised, unsupervised, and reinforcement learning. The goal of supervised learning is to map inputs to labeled outputs, which results in two tasks: Classification (when the output variable takes class labels) and Regression (when the output variable takes continuous values). On the other hand with unsupervised learning one tries to find structures in the input data, which can be achieved in many different ways. Some examples are Clustering (grouping inputs), Projection Methods (reducing dimensionality of the input data), Generative Models, etc. Reinforcement learning instructs the computer to perform a certain goal in a dynamic environment and is provided with feedback in terms of rewards and punishments as the algorithm tries to move through the problem space.

A lot of different learning algorithms already exist, with the most prominent being Artificial neural networks and the extension Deep learning. Furthermore, Support Vector Machines or Decision Trees and its extension random forests, to name a few, have shown their versatility in many fields. The general workflow for training a classifier can be described in 3 main steps: preprocessing, model training, model evaluation. While those 3 steps summarize the workflow pretty well, in reality, a few more steps are necessary to successfully train a robust classifier

(Figure 2.1). Those include the selection of meaningful features and to choose the most fitting machine learning algorithm for the data at hand.

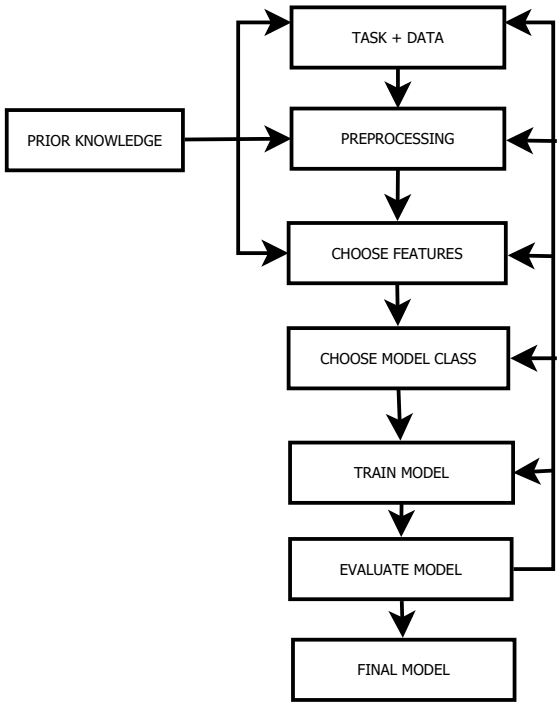


Figure 2.1: Basic data analysis work-flow

2.1 Decision Trees

Decision trees are a natural and intuitive way to classify data because they are analyzing patterns by asking a sequence of questions in which the next question depends on the answer of the current. The answer to these questions is usually

binary, i.e. true/false. In a so-called directed decision tree, the root node represents the first question and is at the top, which is connected via branches to other nodes until a terminal node is reached.

Decision tree learning algorithms are recursive depth first search algorithm that performs hierarchal splits in such a way that leaf nodes are homogeneous (pure) in terms of their target class. Three major question define a decision tree learning algorithm, Which splits should we choose?, When should we stop? How to reduce complexity?.

2.1.1 Splitting Criterion

As a splitting criterion for classification, different kinds of measurements are used, information impurity (information gain) Gini impurity (Gini impurity gain), Variance impurity and misclassification impurity. Let's say i(N) is the impurity of a node N, that means we want i(N) to be 0, if all samples at node N have the same class label and be large if the class labels are uniformly distributed.

The *entropy impurity* of node N is then defined as [25, Chapter 8]:

$$i(N) = -\sum_j \hat{P}(x \in w_j|N) log_2 \hat{P}(x \in w_j|N)$$

Where $\hat{P}(x \in w_j|N)$ denotes the fraction of training samples x at node N that are in the category w_j, given that they survived all previous decisions that led to node N. Now the entropy impurity will be 0 if all the samples at node N belong to the same class and positive if otherwise, with the maximum if all samples are uniformly distributed.
For a given set of training samples $\mathbf{T}$ in the form $(\mathbf{x}, y) = (x_1, x_2, ..., x_n, y)$ where y is the class label the information gain is then defined as:

$$IG(T, a) = i(T) - \sum_{v \in vals(a)} \frac{|\{\mathbf{x} \in T | x_a = v\}|}{|T|} \cdot i(\{\mathbf{x} \in T | x_a = v\})$$

Usually, all possible splits in each step are considered and the one with the highest information gain is selected. This is the standard measurement used in the tree learning algorithms, ID3 [22] and C4.5 [27].

Apart from the entropy impurity, the Gini impurity is also widely used and is defined as [25, Chapter 8]:

$$i(N) = \sum_{i \neq j} \hat{P}(x \in w_i|N) \hat{P}(x \in w_j|N) = 1 - \sum_j \hat{P}^2(x \in w_j|N)$$

The Gini impurity is then just the expected error rate at node N if the category is selected randomly from the class distribution at N. Now the Gini impurity gain for a given dataset $\mathbf{T}$ will be in the same form as the information gain, we just use the Gini impurity instead of the entropy impurity.

Lastly, the misclassification impurity can be written as [25, Chapter 8]:

$$i(N) = 1 - \max_j \hat{P}(x \in w_j|N)$$

Which represents the measurement of the minimum probability that a training pattern would be misclassified at node N.

Gini impurity, Entropy impurity and misclassification impurity are similar to each other, but Gini impurity and Entropy impurity are differentiable and hence more accessible for numerical optimization. Additionally, they are both more sensitive to changes in the node probability compared to the misclassification entropy.

2.1.2 Stopped Splitting

Now that we have seen how decisions are made during the learning phase it is important to know when to stop splitting. On one hand, a full-grown tree cannot be expected to generalize well in a problem having a high irreducible error, and on the other hand, a small tree, i.e. splitting is stopped too early, the error on the training set will be too low, thus the performance might get worse. In other words, deciding when to stop splitting is concerned with finding the best complexity (i.e. under-fitting vs over-fitting).

To solve this problem of finding the right complexity, one can use the traditional approach to parameter tuning, i.e. cross-validation, and continue splitting until the error on the validation data is minimized. Furthermore, one can set a threshold value for the reduction in impurity, i.e. if the best split candidate at a given node reduces the impurity by less than the set threshold. This method bears two major advantages over cross-validation. First, all training samples are directly used for training the tree and second, leaf nodes can be in different levels of the tree, which is preferable. Another simple method is to stop splitting when a node represents fewer than some set threshold of points or some set percentage of all training samples.

2.1.3 Pruning

As an alternative to stopped splitting, *pruning* can be used. In this approach, the tree is fully grown until the leaf nodes have reached minimum impurity. Now all pairs of neighboring leaf nodes are considered for elimination, and whose elimination yield a (small) increase in impurity is eliminated. Two different pruning methods are typically used, cost-complexity pruning and reduced error pruning.

2.2 Tree bagging and random forest

Although Decision trees have many aspects that would make them the most suitable learning method for data mining, e.g. being invariant under scaling and/

or more general transformations, immunity to predictor outliers as well as their inbuilt feature selection, they suffer under one major drawback, their inaccuracy. [11, p.352]

To overcome this drawback, we can make use of the so-called bagging technique or bootstrap aggregation. Such ensemble methods are very common techniques in machine learning since their goal is to reduce inaccuracy, i.e. make more robust models. Bagging draws random samples with replacement (B-times) from a training set and and fits a model to these samples, it averages the prediction over the collection of bootstrap samples, and by that reducing the variance. [11, p.246]

Algorithm 1 Random forest for Regression and Classification [11]

1. For $b = 1$ to B:
   (a) Draw bootstrap sample Z* of size N from training data
   (b) Grow a tree Tb to the bootstrap sample and repeat the following steps for each terminal node until the minimum node size is reached:
      i. Select m variables at random from the p variables
      ii. Pick the best variable among the m
      iii. Split the node into 2 daughter nodes
2. Output the ensemble of trees

As it turns out, trees are well suited for the bagging technique, since they are able to catch complex structures as well as having low bias if grown deeply enough. In addition, each tree generated by bagging is identically distributed so that the bias of bagged trees is identical to that of individual trees and in order to improve the model the variance has to be reduced [11, p.587-588].

Now the idea of random forest (Algorithm 1) is to further improve the variance reduction by de-correlating the grown trees. Correlation between trees in a bootstrap sample can be due to strong predictors, or in other words, if one or more features are very strong predictors for the target, many trees will select these features, causing the trees to be correlated. In order to reduce the correlation of the trees and still have a minimum variance, a method called random subspace method is used at each split to randomly select input variables (subset of features). More specific, for p input variables: Before each split, select $m < p$ of the input variables at random as candidates for splitting [11, p.588] Commonly, $\sqrt{p}$ (rounded down) or even 1 are used as values for m in a classification problem [11, p.588].

2.2.1 Attributes of random forests

Random forests incorporate many attributes that make them extremely viable for data mining and machine learning. These include Out-of-Bag Samples, Variable Importance, their resistance against over-fitting and how they can deal with highly imbalanced class distributions.

2.2.1.1 Out-of-Bag Samples

An attribute that comes with the use of bagging is that it can be used to give estimates of the generalization error. These estimates are done OUT-OF-BAG (OOB). Consider a bootstrap training set formed from one specific training set consisting of observations $Z_i = (x_i, y_i)$. A random forest predictor that is constructed by averaging over only those trees corresponding to bootstrap samples in which Z_i did not appear, is considered the out-of-bag classifier. Thus, the OOB estimate for the generalization error is the error rate of the OOB classifier on the training set [6]. A study by Breiman [5] showed that OOB estimate is as accurate as using a test set equal in size to the training set. Hence, using said estimate removes the need of a set aside test set.

2.2.1.2 Variable Importance

Random forest can rank the importance of a feature in a classification problem in a natural way, since the underlying basis are decision trees, which are easily interpretable. However, since random forests rely on bagging, it is hard to see which variables are important and which are not. Although the interpretability in the sense of a single decision tree is lost, we can still obtain the importance of a single predictor through the summation of the Gini impurity.

As defined above, the Gini impurity measures the impurity of the samples assigned to a certain node (Node purity). Every time a split is made, using a certain feature the Gini-Index for the descendant node is less than the parents one. By measuring how much the Gini-Index decreases in a single tree over all nodes that used that exact feature, one can calculate the importance of the features in that tree. In order to calculate the overall importance of a feature in a forest, just average over the importance value among all trees. Another way of measuring the importance of a variable is the following technique as described in Breimans original paper [6]. During the fitting process of a random forest to the data the OOB error for each data point is recorded and averaged over the whole forest. Now to measure the importance of a certain feature variable, the values of this feature are permuted among the training data and the OOB error is again computed on this new permuted set. The importance is then defined as the average of the difference between the OOB errors before and after the permutation over all

trees. Furthermore, this score is normalized by the standard deviation of the OOB differences. A large score indicates high importance of a certain feature variable. However, both methods of determining variable importance have some drawbacks, as pointed out by Carolin et.al. [31]. If data includes categorical variables with a variety of different levels, random forest tends to be biased in favor of those having more levels.

2.2.1.3 Random forest and imbalanced data

Learning from an imbalanced class distribution can be an issue for a lot of machine learning techniques. Random forest is not an exception to that, since it is designed to minimize the overall error rate, and so it will tend to focus the prediction on the majority class which leads to a poor performance when predicting the minority class. As studies have shown ([15], [12], [9]) one way to counter this issue is to make class priors equal, either by down-sampling the majority class or over-sampling the minority class. Down-sampling seems to have the edge over over-sampling, because over-sampling prunes less and therefore generalizes less than under-sampling [9]. Furthermore, Breiman proposed in a paper [13] 2 new techniques, e.g. Balanced Random Forest and Weighted Random Forest. Balanced Random Forest is inspired by the down-sampling idea and works the following way [13]:

1. Per iteration in random forest, draw a bootstrap sample form the minority class. Then draw the same number of samples from the majority class (with replacement)

2. Produce a classification tree from the data with maximum size, without pruning. The tree is produced with the CART algorithm, except instead of searching through all possible variables only search through a set of m randomly selected variables.

3. Repeat the steps above for a number of times desired and aggregate the predictions of the ensemble and output the final prediction.

Weighted Random Forest essentially follows the idea of cost sensitive learning. RF classifier as well as other machine learning classifiers tend to be biased towards the majority class, hence misclassifying the minority class results in a heavy penalty [13]. That means, each class is given a weight, with the minority class having the larger weight (i.e. higher misclassification cost). The weights are then used to weight the Gini criterion for finding splits and in the terminal node to create a 'weighted majority vote'. The latter means that the weighted vote for a class is the weight for that class times the number of samples for that class at the terminal node. The final prediction is then determined by aggregating the weighted vote from each individual tree.

2.3 Evaluation of Classifiers

When doing classification, objects are classified by a classifier using different features. Usually the classifier is not perfect and makes mistakes, i.e he puts objects into the wrong class. From the relative amount of such miss-classifications one can assess the general performance of a classifier. To do so, the 'true' class has to be known, i.e. we need to know if the prediction was correct or not. An example would be a labor test, where the test subject could be tested positive and it is true, positive and it is false, negative and it's true, negative and it's false. Those four measurements are called, true positive, false positive, true negative and false negative. They are usually represented by a confusion matrix.

From the confusion matrix, two measurements, Sensitivity or True-positive-rate and Specificity or True-negative-rate can be extracted, they are defined as follows:

$$\text{Sensitivity} = \frac{\#\text{True Positives}}{\#\text{True Positives} + \#\text{False Negatives}}$$
$$\text{Specificity} = \frac{\#\text{True Negatives}}{\#\text{True Negatives} + \#\text{False Positives}}$$

Different classifiers require different trade-offs between *Sensitivity* and *Specificity*. For example, we would want to have a Spam filter to not classify good emails as Spam, or in other words, we would want the *Specificity* very high. A common tool to asses the trade-off between *Sensitivity* and *Specificity* is the receiver operator characteristics curve - ROC.

2.3.1 The ROC - curve and the AUC

The ROC graph is a graph in which the true positive rate against the false negative rate is plotted. Hence, a ROC plot illustrates the relative trade-off between benefits (TP) and costs (FP).

The point (0,0) in the lower left corner of the ROC space describes the strategy of never issuing a positive classification. On the other hand, the point (1,1) in the upper right corner represents the strategy of unconditionally issuing positive classifications. Hence, the point (0,1) represents perfect classification. Naturally, one point is better than another if its TP rate is high and FP rate is low, i.e. if it appears in the upper left of the ROC space.

The diagonal that can be drawn between the points (0,0) and (1,1) describes the strategy of randomly guessing, a class. The lower left triangle is usually empty since it would require a classifier that is worse than random guessing. However, the decision space is symmetrical about the diagonal, therefore negating a classifier

would turn all true positives into false negatives and all false negatives into true positives. Hence, any classifier producing a point in the lower right can be negated to produce a point in the upper left.

In order to compare a classifiers performance it is necessary to reduce the two-dimensional illustration (ROC graph) into a single scalar value. A common method is to compute the area under the ROC curve, in short - AUC. The AUC will always have a value between 0 and 1 since it is a part of the unit square. However, no realistic classifier would produce an AUC less than 0.5, because that would make the classifier worse than randomly guessing.

2.3.2 Performance and over-fitting of random forests

In general a random forest classifier is rather robust, but if we take a dataset with a huge number of variables, where the majority of them are irrelevant, the random forest classifier is likely to perform poorly. However, if the number of relevant variables increases, the performance of random forest becomes notably robust to an increase of noise variables [11, p.596]. Another claim made by Breiman et.al. in his original paper [6] is that it is nearly impossible for random forests to over-fit the data. While it is certainly true that it is very unlikely for random forest to over-fit it is not impossible. The average of fully grown trees can end up in a model that induces variance, i.e. the model over-fits [11, p.596]. As shown by Segal [16], by controlling the depth of individual trees in a random forest, a minor performance gain can be achieved. However, using fully grown trees cost merely anything and result in less tuning parameters.

Chapter 3

Survival Analysis

Assessing the effect of a certain intervention on a population is often very important. In clinical trials, for example in heart surgery, where the goal is to find out how many subjects are alive after a certain amount of time after the surgery. Or in cancer therapy, where the goal might be to assess the time until cancer relapses. This time, which starts at a certain point and ends at the occurrence of a given event is called the survival time. The analysis of group data is then called survival analysis [10]. It is widely used in medical/biological studies, or in engineering, where it is used to estimate the lifetime of a machine.

Such analyses often suffer under so-called censored data, more precisely - right censored data. Right censored observation are produced by study subjects who refuse to participate any longer in a study or when a subject does not experience the specified event until the end of the study. Those right-censored observations provide partial information, i.e. the event occurred or will occur sometime after the end of a study. On the other hand, subjects can become part of a study at a later time, i.e. the observation time is much shorter and the specified event may not occur in that short time span. Although censored data only provides partial information, it cannot be omitted [10].

3.1 Kaplan-Meier estimator

The Kaplan-Meier estimator offers a way to analyze such data. The resulting survival analysis is called the Kaplan-Meier survival curve and is defined as the probability of surviving in a given length of time while considering time in many small steps. The analysis used three major assumptions [10]:

1. Censored subjects have the same survival prospects as non-censored subjects

2. Survival probabilities are the same for each subject, i.e. early and late recruited subjects have the same survival probabilities as subjects who are part of the study for the whole specified time.

3. The event happens at the specified time.

The Kaplan-Meier estimate, or *product limit estimate*, involves the successive multiplication of probabilities that an event occurred at certain points in time. Let NL be the number of subjects living at start and ND be the number of subjects that died, then the survival probability at any point in time t is given as:

$$S_t = \frac{NL - ND}{NL} \tag{3.1}$$

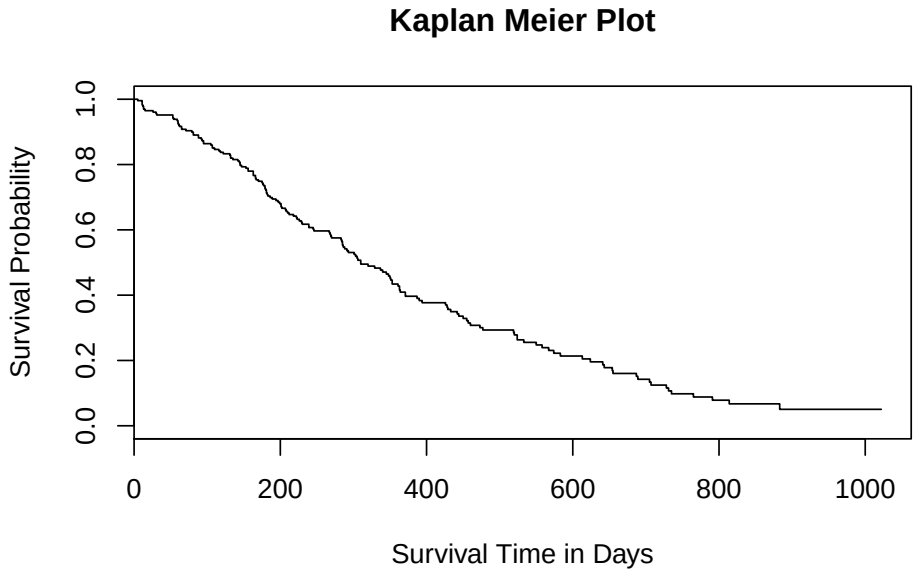
which means the survival probability is given as the number of surviving subjects divided by the subjects at risk. Censored subjects are not counted in the denominator. The total survival probability until a certain time interval is then calculated by multiplying all survival probabilities at all times preceding that time, i.e. calculating the cumulative probability. Although survival probabilities at any time interval are inaccurate, the overall probability of surviving at each point is more accurate.

3.2 Kaplan-Meier plot

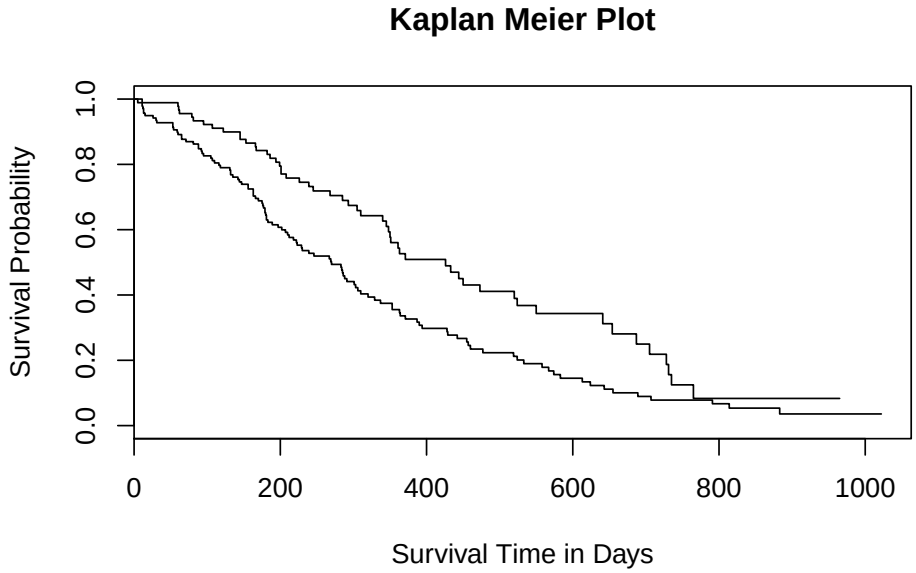
To visualize the Kaplan-Meier estimates, they are plotted as a step function. Such a plot is called Kaplan-Meier plot. On the y-axis, the survival probability is plotted and on the x-axis, the time past after the entry to the study is plotted. The reason a step function is plotted is that the proportions of surviving subjects stay unchanged between the events. A Kaplan-Meier plot for only a single group, as the one shown in Figure 3.1a, is somewhat informative. It is much more interesting to compare survival curves of different groups, e.g. compare the survival percentages of a newer therapy with the standard one (Figure 3.1b). When comparing 2 such curves, one can inspect the vertical gaps or the horizontal gaps. A vertical gap means that at time t one group had a greater fraction of subjects living and a horizontal gap means that it took longer for one group to experience a certain amount of deaths.

3.2.1 Log-Rank Test

It is possible to compare 2 survival curves by looking at the proportions surviving at any given time point. But this approach is flawed, since it cannot provide a comparison of the total survival experience, but instead provides a comparison at



(a) Kaplan Meier plot with 1 group



(b) Kaplan Meier plot with 2 groups

Figure 3.1: Kaplan Meier plots created with the lung dataset from the R - package *survival*

any arbitrary time point. The most common method to compare the survival of groups is the log-rank test, or Mantel-Cox test, which takes the whole follow up period into account. Moreover, it has the advantage that nothing about the shape of the survival curve or the distribution of survival times has to be known [4]. The log rank test statistic L is defined as [10]:

$$L = \frac{(O_1 - E_1)}{E_1} + \frac{(O_2 - E_2)}{E_2} \tag{3.2}$$

where E_1 and E_2 are the expected number of events in the respective group and O_1 and O_2 are the total number of observation in the respective group. The total number of expected events in a group, e.g. E_1, is the sum of expected events at each the time of each event of any group. The expected number of events in any group at the time of an event is given as the product of the risk of an event and the total number of subjects alive at the start of the event in that group. If the total number of events in one group is known, i.e. calculated as the sum of expected events at different times, the total number of events in the other group can be calculated by subtracting the known number of events from the total of observed events in both groups.

The result of the log-rank test is a t-value. Generally speaking, a t-value or t-statistic is a standardized value that is calculated from sample data during a hypothesis test. A t-value of 0 indicates that the sample results exactly equal to the Null hypothesis. But the t-value alone doesn't say much, so in order to be able to interpret the result, the t-value has to to be compared to a $\tilde{\chi}^2$ distribution, which yields a P-value. Depending on the P-value, the Null hypothesis, which says that the two curves are not different from each other can be either rejected or not.

Chapter 4

CASPeR - A Shiny UI For Risk Assessment Of Cardiac Surgeries

CASPeR - Cardiac Surgery Prediction tool, is an application to help doctors do risk assessment of cardiac surgery patients. The tool is written with the R package Shiny. It comes with a full Windows installer for easy installation and requires no prior R or in depth machine learning knowledge. Its foundation is a five step random forest analysis, which consists of loading the data, preprocessing, model training, model evaluation and a survival analysis using a Kaplan-Meier plot. On top sits a Shiny UI which guides the user through the single steps.

4.1 R and Shiny

R [23] is a programming language and environment for statistical computations and graphics. R offers many statistical and graphical packages and is highly extensible. It is often the go to language in statistical research. One of the strengths R possess is the ability to create high-quality plots with ease and the inclusion of mathematical formulas.

The R environment includes a variety of software that makes data manipulation, calculation and displaying graphics easy. For example, it offers a suite of operators for calculation on arrays, especially matrices. It comes with a collection of integrated tool for data analysis as well as graphics facilities to display the data analysis. Finally, the language itself is well developed, simple and effective.

4.1.1 Shiny

Shiny [7] is an open source R - package that provides a powerful web framework to develop web applications using R. A Shiny application consists of two parts: the

user-interface definition and the server logic. This framework is all about making it easy to wire up input values from a web page and make them available for the R code to process and output on the web page again. In consideration of the interactivity that is provided by Shiny applications, the input can change at any time and the output values must be updated immediately to reflect the changes. To do so, Shiny comes with a reactive programming library. Using this library, every time a value that is dependent on a reactive value, or the reactive value itself is changed, the right parts of the code will be re-executed, which in return will cause all depending outputs to update.

Another great feature of Shiny is its extensibility. A variety of packages exist that enhance Shiny apps, e.g. DataTables [35], shinyFiles [21], shinyBS [3], to name a few.

4.2 Architecture

The architecture strictly follows the Shiny framework, though it can be split into four distinct parts, namely data loading, data preprocessing, model training and the analysis of the model and survival curve. The parts resulted directly from the basic machine learning and data analysis routine (Figure 2.1). While only the data loading process consists of more than one action, i.e. loading, cleaning and visualizing the data, the other steps do pretty much what their name suggests. During preprocessing the data is prepared to be used in the model training step, where a model is trained upon the processed data. Finally, the model is evaluated and the resulting prediction is used for performing a Kaplan-Meier analysis.

Moreover, CASPeR is designed to be distributed as an offline standalone application. To do so, Google Chrome Portable and R Portable are used and wrapped together with the CASPeR files into a Microsoft Windows compatible installer.

4.2.1 Loading the data

In order to load data into CASPeR, the R package shinyFiles [21] is used. ShinyFiles allows the user direct access to the file system without 'downloading' the files to a temporary location. To avoid inconsistencies between different file formats, CASPeR is designed to only handle CSV files. Currently 2 different CSV formats are widely used, one uses a comma as a separator between columns and a dot as the decimal point while the other (European) format uses a semicolon as a separator and a comma as decimal point. The latter can be troublesome because R handles commas as separators. By default, CASPeR is set to handle the European CSV format, but this can be easily changed through the UI.

Since CASPeR is limited to CSV files only, data stored in other files, such as Microsoft Excel worksheets, must be converted before it can be used. Unfortunately, the conversion from other formats into the CSV format can sometimes lead to artifacts, which could cause the application to crash. To take care of such artifacts, the data is cleaned before it is processed any further, e.g. whitespaces can cause features to be misinterpreted and are removed.

After cleaning, the data will be checked if the EuroSCORE parameters are included as features. If not, the application will lock itself so it cannot be misused. Otherwise, a matrix of dimension n x 2 is formed, where n is the number of features occurring in the dataset. This matrix will store each feature name and the corresponding type, i.e. if a feature is numerical it will be saved as numerical in this matrix. Throughout the report, this matrix will be refereed to as *typeframe*. Once all the feature types are identified the typeframe will look something like this:

Weight	numerical
ASA	ordinal
stationary retention period	omit

Table 4.1: Typeframe

The matrix is used in two ways. While the typeframe is essential for preprocessing, it goes hand in hand with the data summary. The data summary is designed to help the user find features with falsely dedicated feature types. However the typeframe also serves the purpose of a filter for the data summary.If the user requests a trimmed version of the data summary, the typeframe is used and statistical summaries of relevant features only are displayed.

4.2.1.1 Feature type identification

Identifying each feature's type is an 8 - step process shown in Figure 4.2. Firstly, the EuroSCORE features are extracted and are stored in a separate matrix with their corresponding feature types. Next, each feature is transformed into a factor variable. It will be then checked if it has between 4 and 6 levels or if any value contains a Roman numeral (e.g. II), which was found to be indicative for an ordinal. After extracting and saving the found features, the remaining data is check for binary values. This is a straight forward check, if the feature possesses only 2 levels when transformed into a factor, it can be assumed to a binary variable. Again each found feature is extracted and saved temporarily. Furthermore, the reduced data is checked for numerical features, by simply using the *is.numerical* function from R. Additionally, a dictionary of units (e.g. mmol) is used to even

find features which would be not found through the *is.numerical* function, due to various reasons. Finally, the dataset is checked for categorical features, which is done based on the assumption that categorical values will have a $<$ or $>$ somewhere, or that their name includes a buzzword like *code* or *type*. The remaining features are marked to not be used in the analysis.

After identifying each features type, all that's left to do is to merge the 5 individual matrices into one big one. Final modifications about the feature type assignment through the user interface are done directly on this resulting matrix. Additionally, the typeframe can be saved to a CSV file for later use.

4.2.1.2 Data inspection

Two different methods for inspecting the data are provided. The first one is through the R - package DT [35], which provides an R interface to the JavaScript library DataTables, i.e. R objects such as matrices or data frames can be displayed as tables on HTML pages. Furthermore, the library provides filtering, paging sorting, and many more features. Unfortunately, depending on the size of the data frame rendering the table can be slow. Since Shiny refreshes the web page every time a user input is issued, it can cause the application to become slow. To avoid that, the table is only rendered if the user wants to inspect it, i.e. the user has to press a submit button. In addition, the HTML table produced by DT, a new HTML page can be rendered through an R Markdown [2] file, that offers basic summary statistics of each feature.

Basically, the file takes the data and the currently assigned feature types as inputs. Then checks every feature if it is numeric. If so a table is created, which has to have at most 10 columns, otherwise a summary table of the feature is produced using the *summary* function from R. If the feature is not numeric, a table is created. In either way, the resulting table is converted into an HTML table using the package pander [8]. Unfortunately, the usage of the package pander requires Pandoc to be installed. Pandoc is a universal document converter and the version 1.19.2.1 is distributed with CASPeR.

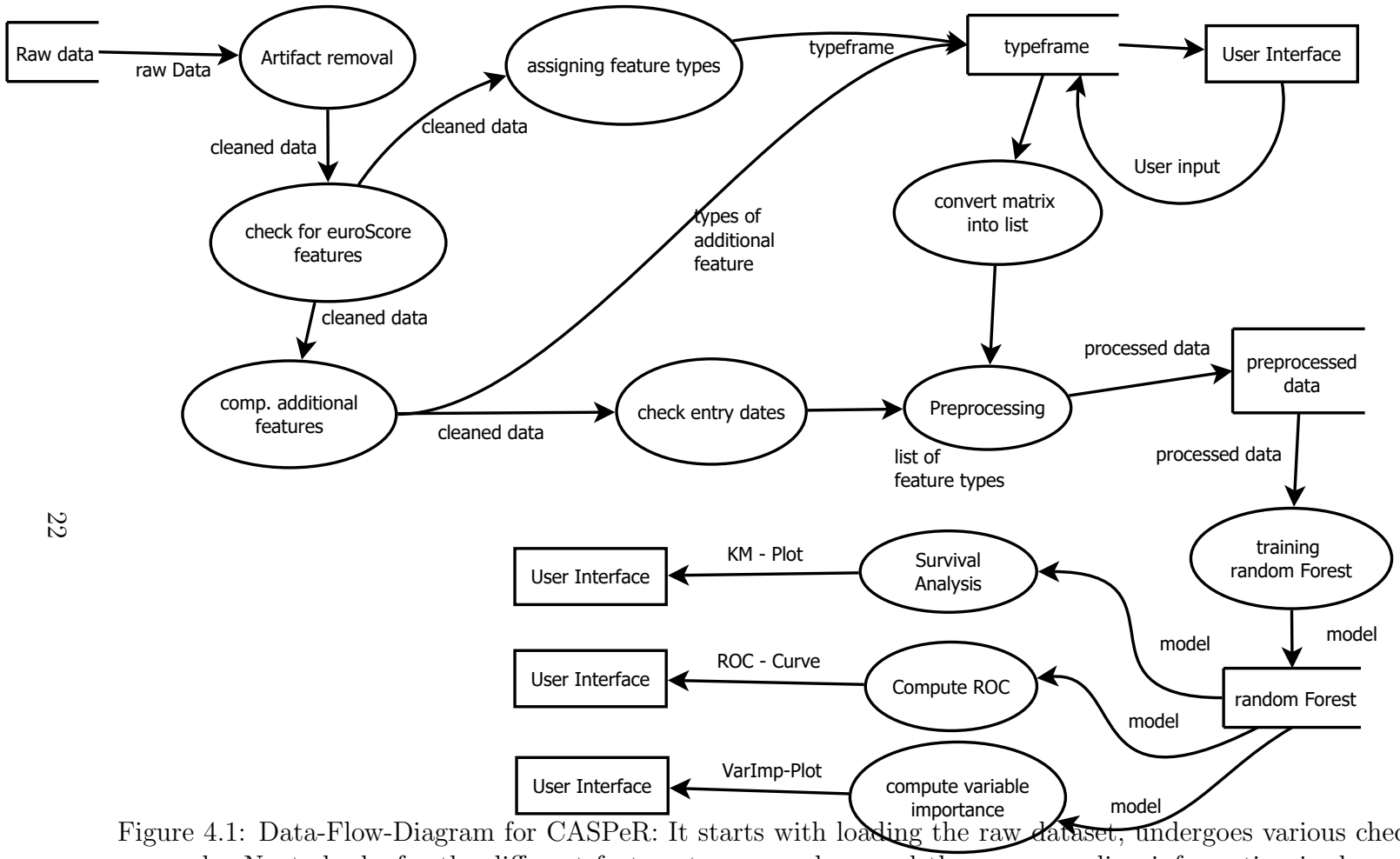


Figure 4.1: Data-Flow-Diagram for CASPeR: It starts with loading the raw dataset, undergoes various checks and removals. Next checks for the different feature types are done and the corresponding information is shown to the user. Then the data is once more processed and a random forest classifier is trained on the processed data. In the end the ROC, AUC, Kaplan - Meier plot as well as the variable importances are shown to the user for further interpretation.

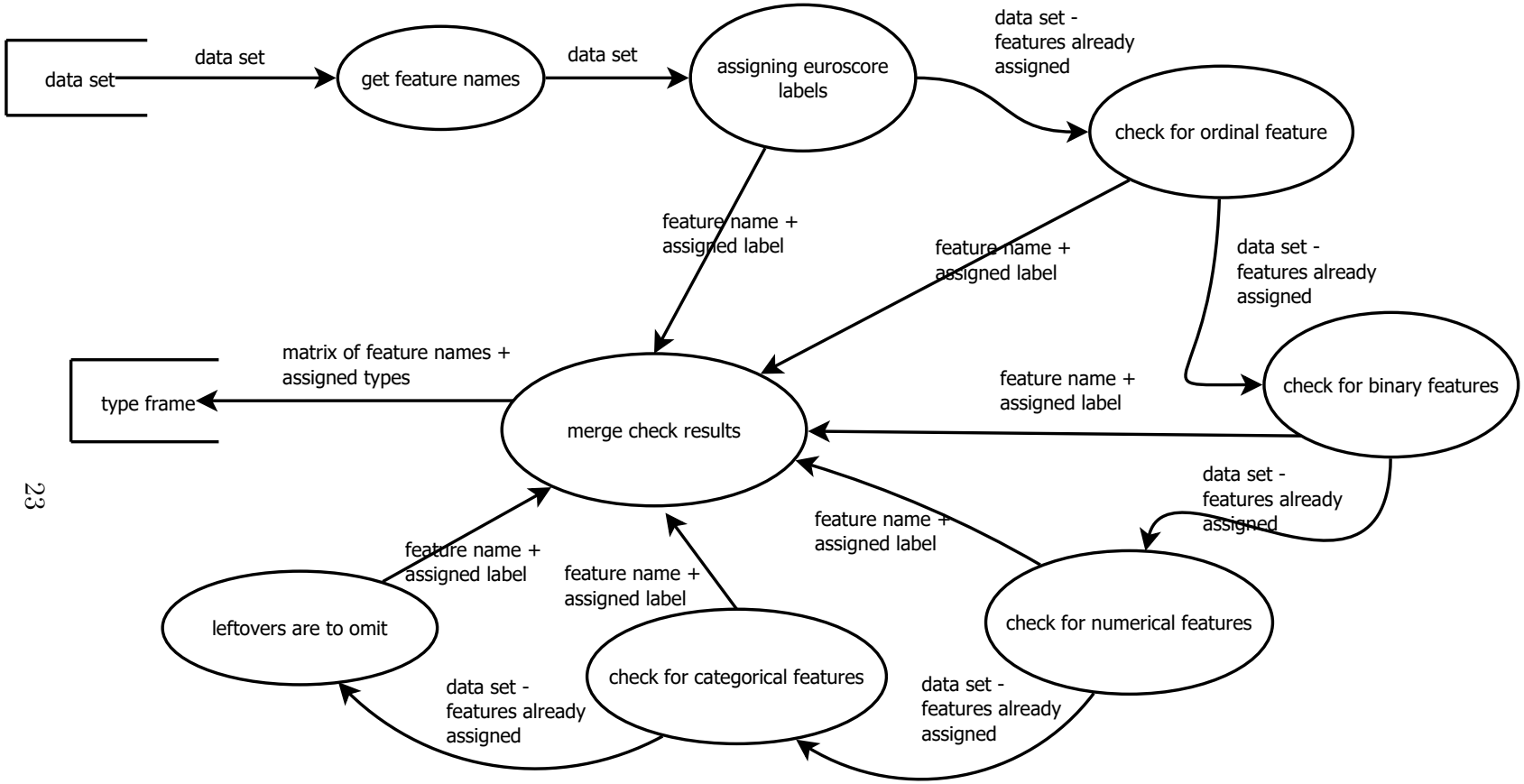


Figure 4.2: Data-Flow-Diagram for the creation of the typeframe: After retrieving the feature names and assigning the EuroSCORE labels, the dataset is subsequently reduced by the check functions and subsets are created, which are then merged into the final typeframe

4.2.2 Preprocessing

Prior to training a model the data usually undergoes extensive preprocessing. Although the data is partly cleaned directly after loading, a more specific algorithm is used to handle missing values and out of range values. In addition, the preprocessing step is used to calculate additional information from a patient entry.

In a first step, the evaluation date of the data entries is checked if they meet the specified survival time, e.g. if the specified survival time is 30 days and the time between the evaluation date and the date of surgery is not 30 than the entry will be excluded from the analysis.

Moreover, the Body-Mass-Index and the postoperative retention period are calculated from the data. The latter is used to calculate the date of surgery and the survival time of an individual patient. If the dataset is missing any information that is needed to perform these steps, and analysis is not possible.

Once the additional information is calculated, the algorithm takes care of the missing values and out of range values. If any feature consists of more than 25% missing values it is excluded. Missing values of numerical features were imputed with the median of non-missing values. Missing values of categorical, ordinal and binary features were imputed with the most frequent values. Besides the treatment of missing values censored numerical data is truncated, e.g. the '< 0.1' entry for CRP is replaced by the limit 0.1.

In a final step, categorical features are encoded as binary features if they had two possible values, and one-hot encoded if they have more than two possible values. Ordinal features are transformed into positive integers.

4.2.3 Training and evaluation of the classifier

The random forest classifier is trained with the random forest implementation in the R - package randomForest [14]. The parameter: survival time, nr. of trees and sampling approach have to be chosen through the UI. The defaults are, respectively, 30 days, 10001 trees and the standard sampling scheme used in the random forest package. Furthermore, the possibility to run the training in a parallelized mode exists. This is realized with the R packages *doSNOW*, *foreach* and *parallel*. If the parallelized mode is selected the number of cores available is detected and displayed to the user. He can then choose the desired number. In a benchmark test, parallelization achieved a 63.54% time reduction using the default sampling, 10001 trees and the dataset described in section 4.5.

However, the number of trees used to train the random forest classifier has to be split by the number of cores used, i.e. the selected number of trees is not equal to the actual number of trees used in the model. The actual number is the number of trees selected divided by the number of cores used and rounded.

In terms of variable importance, only the mean decrease in Gini impurity is saved by the model, while the mean decrease in accuracy is discarded. This yields the advantage of greatly reducing the computational effort. The performance of the trained model as well as the predictiveness of the most important features are given to the user via a ROC plot and the AUC. The computation of the ROC plot and the area under the curve is done with the R - package ROCR [30]. Important to mention is that the evaluation of the performance relies on the Out-of-Bag-Samples (section 2.2.1.1). The use of OOB-Samples also reduces the computational effort compared to N-fold cross-validation.

4.2.4 Survival Analysis

The Survival analysis is concerned with splitting the prediction resulting from the trained random forest model into the target groups, i.e. *alive* and *dead*. Then a survival curve is fitted to produce the Kaplan-Meier plot. The time axis of the Kaplan-Meier plot is variable, which means the user can select what time frame he wants to inspect. The survival analysis is done with the implementations provided by the R - package *survival* [33].

To compare the 2 survival distributions, a log-rank test is done which is implemented with the *survdiff* function in the R - package *survival*. Unfortunately, R rounds numbers to 53 binary digits accuracy, which means that if a p-value reaches this limit (1e-16) it is rounded to 0.

4.3 Distribution of the application

Having a Shiny application distributed as a desktop application bears two major advantages:

- The user does not need any R knowledge to run the app
- The application installer can simply be downloaded

To make a Shiny app a local desktop application, a deployment skeleton is needed. The backbone of this skeleton build R Portable (has all the necessary packages installed), Google Chrome Portable. Chrome is run in *app mode* which makes it look more like a native app. Finally, only two scripts are needed to transform the Shiny app into a standalone application. More precisely, a script that calls R portable non-interactively from where the second script is run to actually launch the application. The final folder structure will look like this:

- C:\App\GoogleChromePortable

- C:\App\RPortable
- C:\App\Shiny
  - ui.R
  - server.R
- C:\App\run.vbs
- C:\App\runShinyApp.R

Now, all that's left to do is to compress it into a .zip archive and distribute or for a more advanced solution InnoSetup [1] can be used. InnoSetup essentially wraps the whole directory into a .exe file which acts as a setup wizard.

4.4 App Usage

When used for the first time it is advisable to set a working as well as a saving directory. The latter is where the feature type matrix is saved to and loaded from, in case the user wants to do so. Once the working and the saving directory are chosen, the application must be restarted for the changes to take effect. The default setting for the working directory and result directory is 'C:\'.After a working directory and a result directory are specified the user can start with the analysis. The general work flow consists of five steps:

1. Loading the data into the App and inspecting it either through the user interface directly or through the summary method. Depending on the CSV style the user can choose either one of the predefined ones (EU or Standard) or select the loading options fitting the CSV file.

2. After the data is loaded the user can proceed and give the feature types a finishing touch, i.e. he can reassign falsely assigned types of features through the user interface. It is recommended to save the assignment from time to time, to avoid losing the already done work in case of system failure or application failure.

3. Once the user has assigned the correct type to each feature, he can proceed to fill in the required fields, e.g. Weight, Height, etc. Once the user has filled in the fields, the **Train** button will be unlocked and the user can proceed to specify the number of trees, if the analysis should be parallelized (if yes, how many cores should be used), which sampling scheme should be used and the desired survival time.

4. Subsequently to training the classifier the performance can be evaluated by inspecting the ROC curve and the AUC value. Besides the performance, the survival curve from the resulting prediction of the classifier is displayed for inspection through the user. In addition, the log-rank test result is shown.

5. Lastly, the user can inspect the most important features according to the mean decrease in Gini impurity displayed by a variable importance plot. The 10 most important features as well as their predictiveness via a ROC graph, are given.

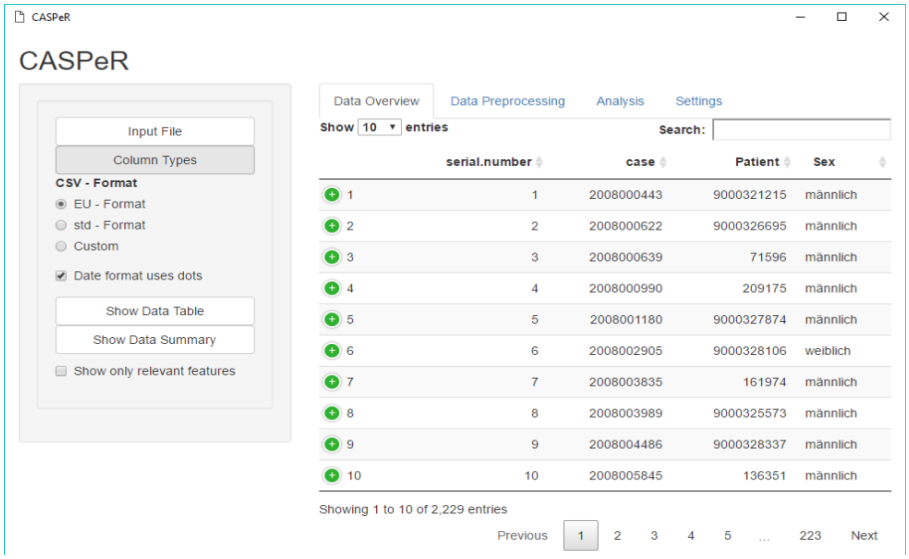
4.4.1 UI - Description

The user interface of CASPeR consists of 4 main tabs, where three are concerned with the analysis and one is concerned with the settings of the application. The first tab, *Data Overview*, is used for loading the data and to gain some general information about the data. The second tab, *Data Preprocessing*, is as the name already indicates, concerned with handling the input necessary to preprocess the data as described in section 4.2.2 and 4.2.2. In the third tab, *Analysis*, performance evaluation, a Kaplan-Meier curve and the most important variables are shown to the user.

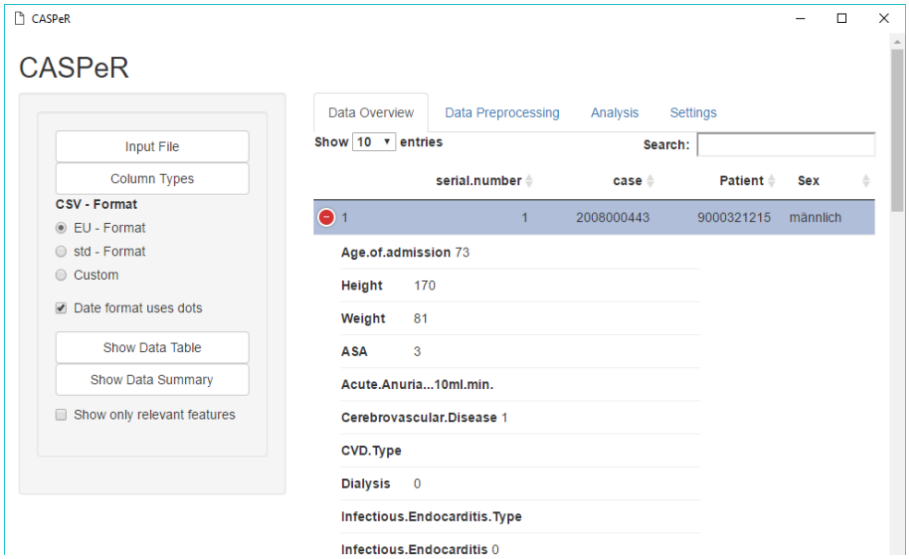
4.4.1.1 Data Overview

The side panel of the *Data Overview* tab (Figure 4.3a) holds the option to load the data file and the file indicating the feature types. When pressing the **Input File** button the directory specified as the working directory is opened and all CSV files in this directory are shown for selections. Once a file is selected it is loaded. To inspect the data table the **Show Data Table** button must be pressed. Afterwards, the table is presented, where each row corresponds to a complete patient record. On the left side of the table, a green plus can be found. After invoking it the table will expand and show all clinical parameters of the selected patient record (Figure 4.3b). To collapse the menu again, the red minus must be invoked. Moreover, the user can select how many records should be displayed by using the *Show 10 entries* option in the top left corner of the table. If the user feels the need to directly search for an entry, he can do so by using the *Search* function in the top right corner (Figure 4.3a). Finally, to switch to next page of entries the slider at the bottom of the table can be used.

Directly below the buttons **Input File** and **Column Type**, loading options for the dataset can be found. The user can choose between two predefined options as well as the option for custom loading parameters. The predefined options are for the European and Standard CSV style. The custom option gives the user access



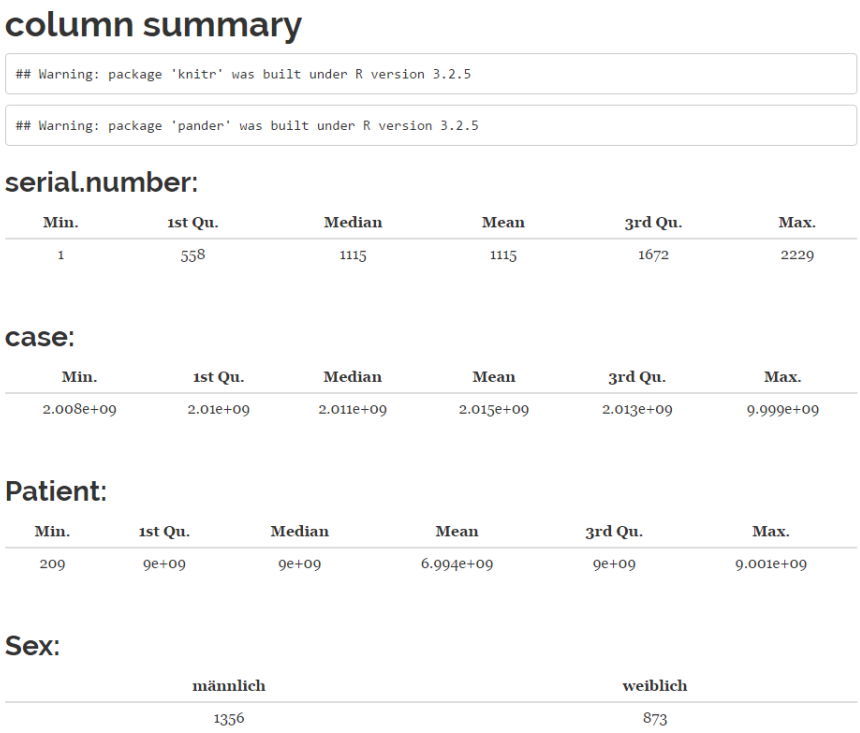
(a) Data Overview tab with data table



(b) Expanded data table

Figure 4.3: Input File & Column Types: Load data file into the application. EU-Format, std-Format, Custom: Choose parameters for loading the data file. Show Data Table: Display an overview of the data. Show Data Summary: Open a new browser with a basic statistical summary of the data. Show only relevant features: Exclude features not used in the analysis, from the statistical summary.

to change the following parameters: Separator, Quote, Decimal Mark, headline or not. Moreover, the user must tell the application if the date format uses dots or not. He can do so through the checking or unchecking the box below the button for custom loading parameters. The lower section of the side panel carries the buttons **Show Data Table** and **Show Data Summary** as well as the option *Show only relevant feature*. As described above the **Show Data Table** will display the data table, but after invoking the **Show Data Summary** button and a short wait time a browser window is opened with a web page displaying summary statistics of each clinical parameter existent in the dataset (Figure 4.4). The user can decide, whether he wants to show all summary statistics for all parameters or only the relevant ones, i.e. all features which are initially assumed to be relevant for the analysis.



column summary

Warning: package 'knitr' was built under R version 3.2.5

Warning: package 'pander' was built under R version 3.2.5

serial.number:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	558	1115	1115	1672	2229

case:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
2.008e+09	2.01e+09	2.011e+09	2.015e+09	2.013e+09	9.999e+09

Patient:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
209	9e+09	9e+09	6.994e+09	9e+09	9.001e+09

Sex:

männlich	weiblich
1356	873

Figure 4.4: Small part of the data summary

4.4.1.2 Data Preprocessing

The tab **Data Preprocessing** holds the input field for required input parameters, e.g. the names of the weight and height column, number of days after which the mortality should be predicted, etc. (Figure 4.5a). Each of those fields is required

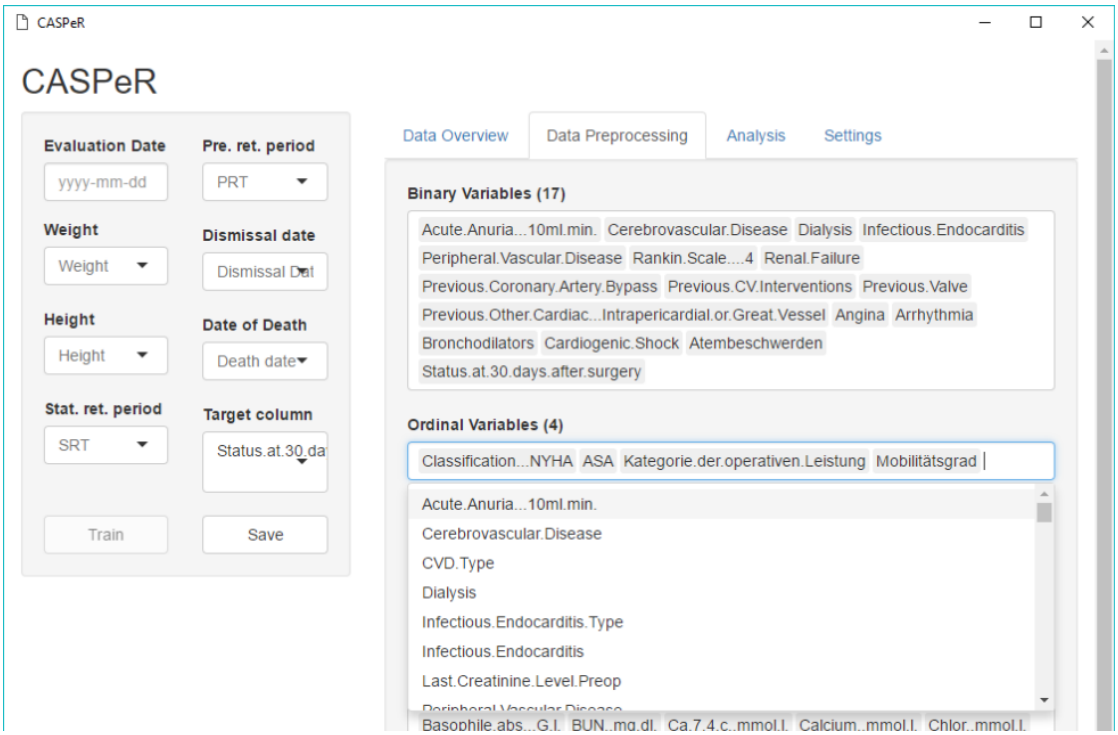
to be filled out or otherwise the classifier training is not possible. The *Evaluation Date* refers to the date when the dataset was obtained from a database. The input format has to be in the following format: YYYY-MM-DD. For the fields *Weight, Height, Stat.- Pre. Retention period, Discharge Date and Date of Death*, the corresponding column names in the dataset must be specified. Last a target column can be specified. By default, it is set to the prediction of the mortality after a specified amount of days. If all fields are filled in, the **Train** button will be unlocked. After pressing a new window will appear which handles all parameters of the model. Additionally, the user can specify if he wants to run the analysis in parallel mode or serial mode (Figure 4.5b).

The main panel holds five fields corresponding to the five feature type *ordinal, binary, categorical, numerical, omit*. Each field holds elements named after features. To reassign one feature type, the user can select a field and type in the name of the feature he wants to assign to the certain type. A drop down menu will appear (Figure 4.5a). The selected feature name is then removed from the field it before was and added to the selected field. If a feature element itself is selected via mouse click and deleted via the Backspace key or the Delete key, it is automatically moved to the *omit* field. The application will refresh the whole page after each change, due to Shinys reactive programming library.

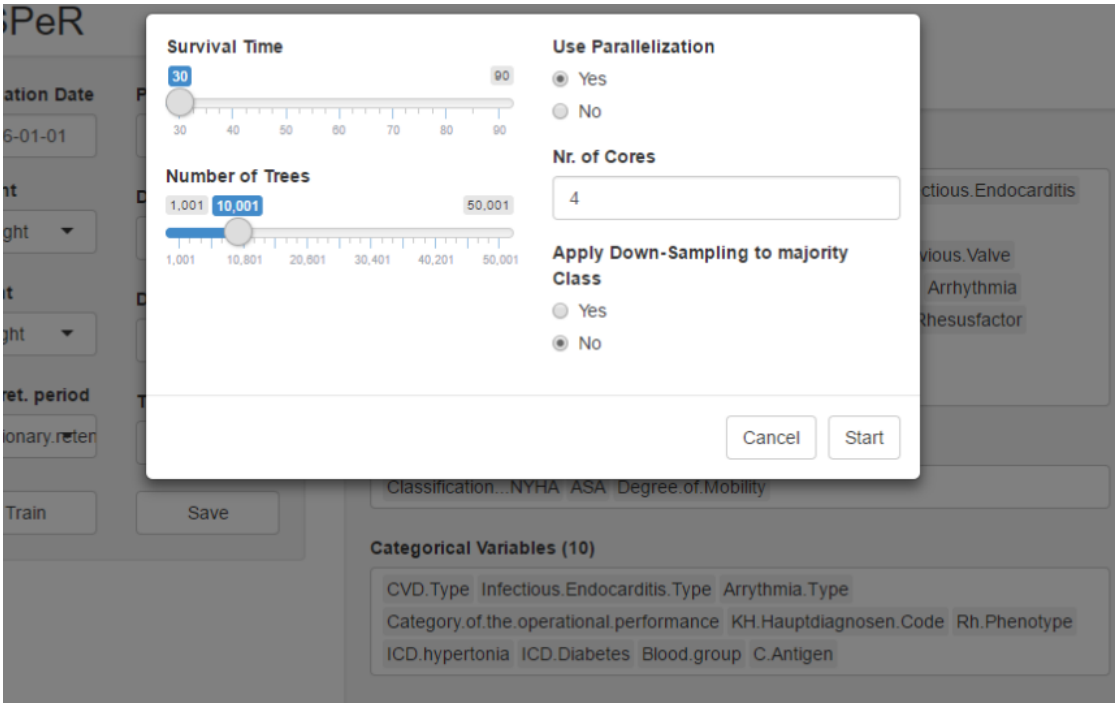
4.4.1.3 Analysis

The last section used for the analysis only consists of 2 main fields. Three radio buttons located in the upper left, and a plot area in the main panel. Depending on what is selected through the radio buttons, the according plot will be displayed, i.e. if the Kaplan-Meier analysis is selected the Kaplan-Meier curve is displayed.

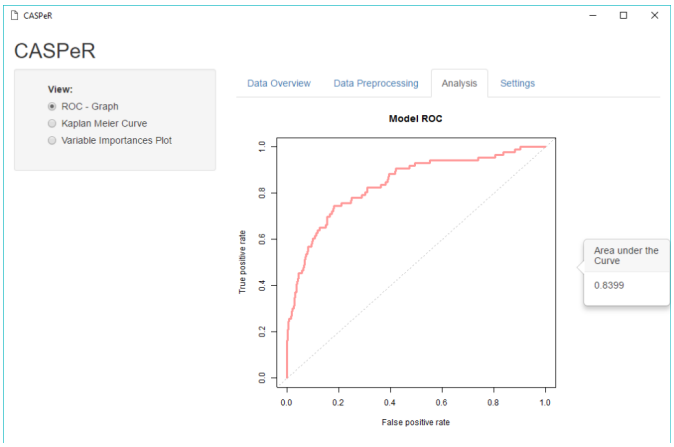
The ROC plot's special feature is that if the user hovers the mouse pointer over the plot, a small window will appear displaying the area under the curve value (Figure 4.6a) If the selection is switched from *ROC* to *Kaplan-Meier*, the plot will change and the Kaplan-Meier curve for the predicted samples is shown. Besides the plot, a slider will appear in the side panel, which is used to scale the time axis (Figure 4.6b). Finally, if the selection is switched from *Kaplan-Meier* to *Variable Importances*, the plot will switch once again into a variable importance plot regarding the mean decrease in Gini impurity. As described in section 2.2.1.2 a useful variable will tend to split the mixed labels into rather pure classes, meaning it will give a relatively large decrease in Gini impurity, i.e. the variable will be ranked high in the plot. The 10 most useful features are displayed (Figure 4.6c). The button **Show predictiveness** opens a new window which shows the ROC curve for each important feature. If the the user hovers the mouse pointer over the plot the AUC value will be displayed.



(a) **Left:** Selection fields for preprocessing parameters. **Right:** Feature type fields and dropdown menu for selecting a feature to assign to a new type.

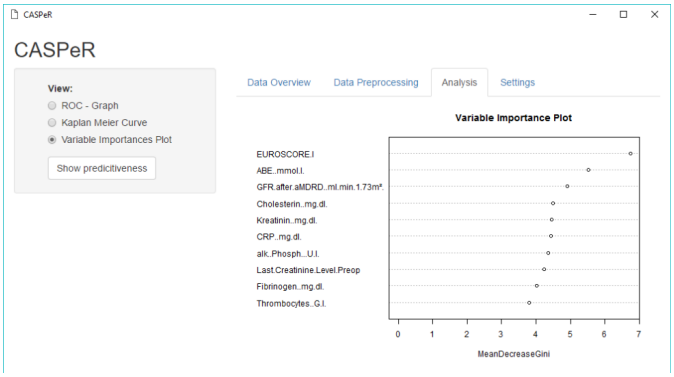


(b) Parameters for model training.



(a) ROC curve with the according area under the curve.

(b) Kaplan Meier plot with adjustable time axis + log-rank test result.



(c) Variable importance according to mean decrease in Gini impurity

4.4.1.4 Settings

The settings tab fulfills the sole role of setting a working directory and directory for saving. Two text fields will display the currently selected directory paths (Figure 4.7). Besides, a button is below each of them which opens a window similar to the one opened by the *Input Files* button, but instead of files the user chooses a directory. Unfortunately, the window responds quite slowly to user input.

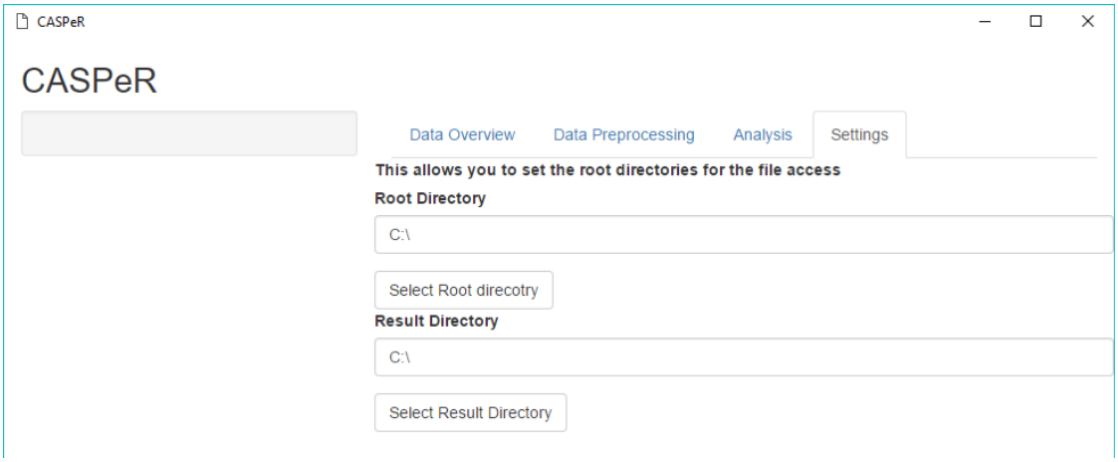


Figure 4.7: Settings tab with the working and saving directory selection.

4.5 Case study

After showing the foundation of CASPeR, architecture and UI layout, we shall now use CASPeR to experiment with a real world dataset. Over the course of 6 years (01.01.2008 - 31.12.2014) 2253 patients underwent heart valve surgery at the Kepler University Clinic Linz. After excluding 24 record due to different exclusion criteria like, re-operation of the same patient, patients that need anterograde cerebral perfusion due to aortic arch surgery and patients with 'grown up congenital heart disease' (GUCH). From the remaining 2229 patients data was collected. All patient data was merged from different subsets 'Crystal Reports' (SAP^{TM}) and anonymized. The data included, age, weight, height, gender, ASA score, all parameters obtained from A-IQI Austrian Impatient Quality Indicators, all preoperative laboratory values, and the EuroSCORE I. In total, each record contained 139 structural parameters. About 37% of the data consisted of missing values. A small demographic overview, obtained from the data summary function is given in 4.2.

The prediction target of this experiment was the 30-days mortality. The mortality rate within the 30-days after the surgery was 3.86%.

Age (years)	68
Height (cm)	169.2
Weight (kg)	78
Male (%)	60.8
Female (%)	39.2
patients with cerebrovascular disease (%)	11.6
patients with peripheral vascular disease (%)	8.0
patients with endocardities (%)	5.1
patients with angina pectoris (%)	26.6
patients with arrythmia (%)	22.7
glomerular filtration rate (ml/min/1.73m)	74
BUN (mg/dl)	21.7
Kreatin (mg/dl)	1.126

Table 4.2: Overview of preoperative data

After careful inspection of the data, 18 columns were omitted due to redundancy or correspondence to intra-operative or post-operative features. The remaining 124 features were then used to train two random forest models. Model A was trained using the standard sampling scheme, which takes the total number of samples, but sampling is done with replacement. Therefore, Model A is optimized for accuracy, while Model B will be trained with the Balanced Random Forest sampling scheme which optimizes for balanced accuracy. More details can be found in Section 2.2.1.3. Each random forest was constructed as an ensemble of 10.001 trees, which ensures relatively robust and accurate models.

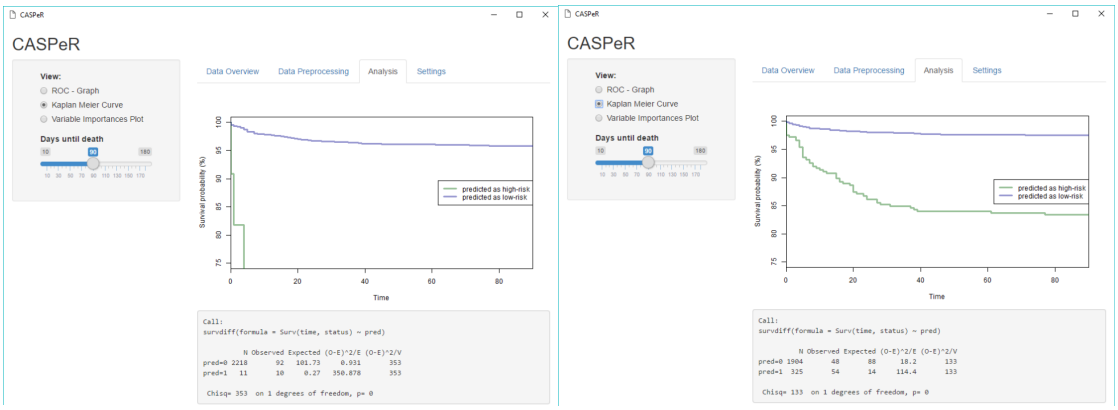
Both models yield good results in terms of AUC. Model A did slightly better than Model B with an AUC of 0.8399 vs. 0.804. Which is only logical because Model A optimizes for accuracy. When investigating the importance of features in the two models we observe the EuroSCORE I as either the highest (Model A) or second highest (Model B) ranked. Other features both models include are the Glomerular Giltration Rate (GFR), Creatine, Last Creatine Level Preoperative, Calcium. If we consider the individual ranking performance, we see that even though EuroSCORE I is ranked highest in Model A it performs worse than GFR with an AUC of 0.7044 vs 0.7409. However, Model B produces the same AUC values for both features but ranks them differently. Furthermore, we observe generally lower AUC values for the top 10 features in Model A than in Model B. The AUC values for the individual features are given in Table 4.3.

Noticeably ABE is ranked as the most important feature after EuroSCORE I, although having a much worse ranking performance than GFR for example. This seems to contradict the fact that ABE is ranked so high by the model. The

Model A	AUC	Model B	AUC
EuroSCORE I	0.704	GFR	0.7409
ABE	0.5192	EuroSCORE I	0.704
GFR	0.7409	BUN	0.7024
Creatine	0.675	Creatine	0.675
Cholesterin	0.5535	proBNP	0.6553
alcalic phosphatase	0.5748	Last creatine level preop	0.6499
Last creatine level preop	0.6499	Erythrocytes	0.6595
Fibrinogen	0.5507	Weight	0.6241
Thrombocytes	0.5418	Height	0.6161
Calcium	0.6304	Hemoglobin	0.6664

Table 4.3: AUC for individual ranking performance of most important features.

ROC - curve clears up this contradiction: the straight incline of the ROC curve in the bottom left corner (Figure 4.8) indicates that high ABE values are observed for patients who die within the first 30 days after surgery. Hence, if we split of these high values, we would correctly classify about 10% of the deceased patients. All other risk parameters including the EuroSCORE I, can be understood as 'the higher the score the higher the risk'.



(a) Survival curve for Model A

(b) Survival curve for Model B

Figure 4.9: Survival curves for both Models

The question, what the mortality rate for a high-risk and low-risk is given our experiment, remains open. To find an answer, we shall investigate the survival curves (Figure 4.9). The survival curve for Model A is obviously much sharper, because the model is much more precise/accurate and predicts 11 patients as deceased while only 10 are actually dead. The survival curve for Model B on the

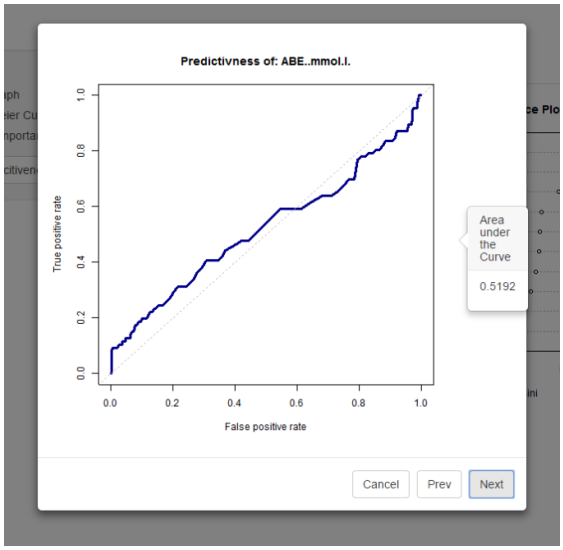


Figure 4.8: ROC curve for the feature ABE. Notice the sharp incline in the lower left corner.

other hand, can tell us much more about the mortality rate, since the model is optimized for balanced accuracy. The mortality rate in the first 30-days after surgery for patients in the high-risk group is 13.89% (46 out of 331). The experiment showed some interesting result. On one hand it is not obvious which sampling scheme would fit best, but both produce very good AUC values. On the other hand, both models trained only had 1 EuroSCORE parameter as a parameter of importance (excluding EuroSCORE I). Moreover, the results shows that kidney-related parameters are of high importance for the model. The Glomerular Filtration Rate even showed a better prediction performance than EuroSCORE I. Besides the highly interesting result, the experiment somewhat showed how simple it is to carry out the such an analysis. Although, the feature type assignment spawns a rather inconvenient step during the analysis, once it is done, saving allows reusing the assignment provided that the future dataset contains the same features.

Chapter 5

Conclusion

Adequate risk profiling and predicting the outcome of a surgery are necessary to provide high-quality information for patient and doctors alike. In the context of consent of the patient, it is of crucial importance to discriminate high-risk patients from low-risk patients. Yet clinicians are provided with a collection of different predictive models to choose from. The three most prominent ones are: the EuroSCORE I and II [18, 19], the STS score [28, 20, 29] and the ACEF (Age, Creatine Ejection Fraction) [24]. Each model comes with its own advantages and disadvantages, although a meta-study has found that the EuroSCORE II and the STS score outperform the ACEF score on discrimination [32].

Nonetheless, all three models yield on major disadvantage compared to machine learning methods: when one of the models is used on data that differs from the one that was used for development, the results will differ [32]. This could be overcome with modern machine learning methods. Especially random forest, seems to be the go-to choice, due to its integrated feature selection, immunity against over-fitting and how it can deal with imbalanced class distributions. Moreover, a Kaplan-Meier analysis offers a great way to analyze the resulting prediction.

Unfortunately, machine learning methods are usually hidden behind programming languages like R or Python. Luckily, the R package Shiny makes it possible to develop outstanding applications, that can offer statistical analysis procedures to people without any background in R. Availability of great additional packages such as DT, shinyFiles, and the option to run the application locally with R Portable and Google Chrome Portable, enhances the user experience further. Although R does not possess the greatest computational performance compared to other programming languages, a package like Shiny makes it the go-to choice for a project like CASPeR. Moreover, the computational performance can be bettered by parallelizing certain routines, like the model training procedure. The reactive programming library which keeps the application updated upon each user input, and the option to use CSS for customization, support the development of nice

looking and interactive apps. By keeping the user interface of CASPeR closely to the general workflow of data analysis, it gives the user a better feeling of what is going on. Admittedly, the feature type assignment could be designed in a way such that it doesn't seem overwhelming if the number of features increases.

Moreover, by enforcing certain features, the improper use of CASPeR is prevented. Although, after installing, the source code can be altered through a simple text-editor but this would require in-depth knowledge about R and would render the aim of this project pointless. Distributing the application as a standalone desktop app was the right choice since it facilitates offline use, i.e. no uploads of large datasets to a server are required. On the other side, a desktop application written in R that uses a browser to display the analysis brings a natural disadvantage: cyber security software. For example, McAfee^{TM} has shown to greatly worsen the performance or even hinder CASPeR from starting. Unfortunately, shutting down the Firewall does not solve this problem, because it seems that some *Validation process* which cannot be shut down, is causing the issue. Up to date, a solution to this problem could not be found.

In conclusion, CASPeR offers a predictive analysis pipeline, that gives doctors detailed information about patients risk profiles with the help of random forest. CASPeR does not require any expertise in machine learning or statistics and is simple to use, and yet produces high-quality results on real-world datasets (AUC: 0.8399). With more research, it might be possible to extend the use of CASPeR to other surgical data.

Bibliography

[1] Innosetup homepage. `http://www.jrsoftware.org/isdl.php`.

[2] JJ Allaire, Joe Cheng, Yihui Xie, Jonathan McPherson, Winston Chang, Jeff Allen, Hadley Wickham, Aron Atkins, and Rob Hyndman. *rmarkdown: Dynamic Documents for R*, 2016. R package version 1.1.

[3] Eric Bailey. *shinyBS: Twitter Bootstrap Components for Shiny*, 2015. R package version 0.61.

[4] J. M. Bland and D. G. Altman. The logrank test. *BMJ*, 328(7447):1073, May 2004.

[5] Leo Breiman. Out-of-bag estimation. 1996.

[6] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[7] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. *shiny: Web Application Framework for R*, 2016. R package version 0.14.1.

[8] Gergely Darczi and Roman Tsegelskyi. *pander: An R Pandoc Writer*, 2015. R package version 0.6.0.

[9] C. Drummond and R Holte. C4.5, Class Imbalance, and Cost Sensitivity: Why Under-sampling beats Over-Sampling. *Proceedings of the ICML-2003 Workshop: Learning with Imbalanced Data Sets II*, 2003.

[10] M. K. Goel, P. Khanna, and J. Kishore. Understanding survival analysis: Kaplan-Meier estimate. *Int J Ayurveda Res*, 1(4):274–278, Oct 2010.

[11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*. Springer, 2nd edition, 2016.

[12] M. Kubat and S. Matwin. Addressing the curse of imbalanced data sets: One-sided sampling. *Proceedings of the 14th International conference on Machine Learning*, pages 179–189, 1997.

[13] Chen Chao & Liaw Andy & Breiman Leo. Using random forest to learn imbalanced data. Discussion paper, Department of Statistics,UC Berkeley, 2004.

[14] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[15] C. Ling and C. Li. Data Mining for direct marketing problem and solutions. *Proceedings of the fourth International Conference on Knowledge Discovery and Data Mining, New York*, 1998.

[16] Segal M. Machine learning benchmarks and random forest regression. *Technical report, eScholarship Repository, University of California*, 2004.

[17] S. A. Nashef, F. Roques, B. G. Hammill, E. D. Peterson, P. Michel, F. L. Grover, R. K. Wyse, and T. B. Ferguson. Validation of European System for Cardiac Operative Risk Evaluation (EuroSCORE) in North American cardiac surgery. *Eur J Cardiothorac Surg*, 22(1):101–105, Jul 2002.

[18] S. A. Nashef, F. Roques, P. Michel, E. Gauducheau, S. Lemeshow, and R. Salamon. European system for cardiac operative risk evaluation (EuroSCORE). *Eur J Cardiothorac Surg*, 16(1):9–13, Jul 1999.

[19] Samer A.M. Nashef, Franois Roques, Linda D. Sharples, Johan Nilsson, Christopher Smith, Antony R. Goldstone, and Ulf Lockowandt. Euroscore ii. *European Journal of Cardio-Thoracic Surgery*, 41(4):734, 2012.

[20] S. M. O'Brien, D. M. Shahian, G. Filardo, V. A. Ferraris, C. K. Haan, J. B. Rich, S. L. Normand, E. R. DeLong, C. M. Shewan, R. S. Dokholyan, E. D. Peterson, F. H. Edwards, and R. P. Anderson. The Society of Thoracic Surgeons 2008 cardiac surgery risk models: part 2–isolated valve surgery. *Ann. Thorac. Surg.*, 88(1 Suppl):23–42, Jul 2009.

[21] Thomas Lin Pedersen. *shinyFiles: A Server-Side File System Viewer for Shiny*, 2016. R package version 0.6.2.

[22] J.R Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[23] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

[24] Marco Ranucci, Serenella Castelvecchio, Lorenzo Menicanti, Alessandro Frigiola, and Gabriele Pelissero. Risk of assessing mortality risk in elective cardiac operations. *Circulation*, 119(24):3053–3061, 2009.

[25] David G. Stork Richard O. Duda, Peter E. Hart. *Pattern Classification, 2nd Edition*. Wiley-Interscience, 2 edition, 2000.

[26] F. Roques, S. A. Nashef, and P. Michel. Risk factors for early mortality after valve surgery in Europe in the 1990s: lessons from the EuroSCORE pilot program. *J. Heart Valve Dis.*, 10(5):572–577, Sep 2001.

[27] Steven L. Salzberg. C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16(3):235–240, 1994.

[28] D. M. Shahian, S. M. O'Brien, G. Filardo, V. A. Ferraris, C. K. Haan, J. B. Rich, S. L. Normand, E. R. DeLong, C. M. Shewan, R. S. Dokholyan, E. D. Peterson, F. H. Edwards, and R. P. Anderson. The Society of Thoracic Surgeons 2008 cardiac surgery risk models: part 3–valve plus coronary artery bypass grafting surgery. *Ann. Thorac. Surg.*, 88(1 Suppl):43–62, Jul 2009.

[29] D. M. Shahian, S. M. O'Brien, G. Filardo, V. A. Ferraris, C. K. Haan, J. B. Rich, S. L. Normand, E. R. DeLong, C. M. Shewan, R. S. Dokholyan, E. D. Peterson, F. H. Edwards, and R. P. Anderson. The Society of Thoracic Surgeons 2008 cardiac surgery risk models: part 1–coronary artery bypass grafting surgery. *Ann. Thorac. Surg.*, 88(1 Suppl):2–22, Jul 2009.

[30] T. Sing, O. Sander, N. Beerenwinkel, and T. Lengauer. Rocr: visualizing classifier performance in r. *Bioinformatics*, 21(20):7881, 2005.

[31] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(1):25, 2007.

[32] P. G. Sullivan, J. D. Wallach, and J. P. Ioannidis. Meta-Analysis Comparing Established Risk Prediction Models (EuroSCORE II, STS Score, and ACEF Score) for Perioperative Mortality During Cardiac Surgery. *Am. J. Cardiol.*, 118(10):1574–1582, Nov 2016.

[33] Terry M Therneau. *A Package for Survival Analysis in S*, 2015. version 2.38.

[34] S. F. Weng, J. Reps, J. Kai, J. M. Garibaldi, and N. Qureshi. Can machine-learning improve cardiovascular risk prediction using routine clinical data? *PLoS ONE*, 12(4):e0174944, 2017.

[35] Yihui Xie. *DT: A Wrapper of the JavaScript Library 'DataTables'*, 2016. R package version 0.2.

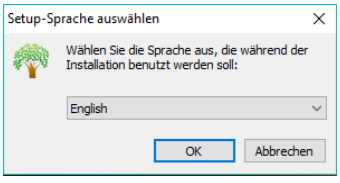
Appendices

Appendix A

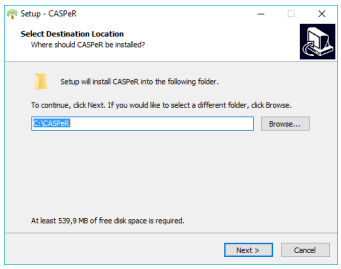
Installation

The installation under Windows 10 is as follows:

1. At first a prompt will ask if the installer should be executed.
2. Choose your preferred language (English/German).(A.1a)
3. Select the destination location where CASPeR should be installed.(A.1b)
4. Decide whether a shortcut should be created or not and if pandoc 1.19.2.1 should be installed. Preferable location: 'C:\CASPeR'. Any other location than 'C\Program Files' is fine.(A.2a)
5. The application is ready to install.(A.2b)
6. Installation progress.
7. The setup is now finished. Depending on whether you installed Pandoc or not a system restart might be required.(A.3)

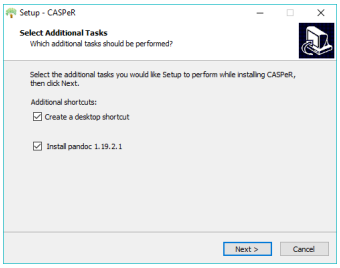


(a) Selecting a preferred language.

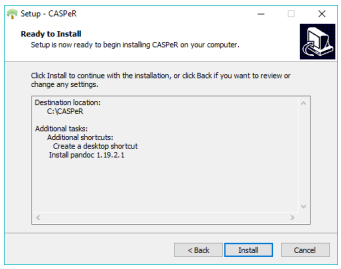


(b) Select Destination Location

Figure A.1: Language selection and installation destination

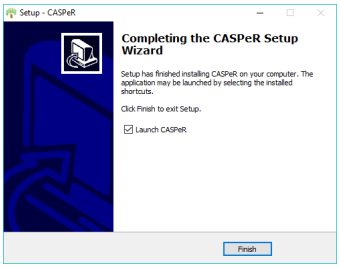


(a) Creation of desktop shortcut and installation of pandoc.

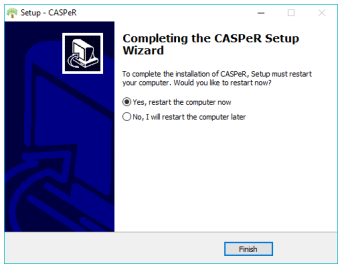


(b) Installation is ready

Figure A.2: Final steps before installation



(a) Installation is finished



(b) Restart is required. Click Finish

Figure A.3: Installation finish with and with pandoc installation