Ludwig-Maximilians-University
Munich

Institute for Statistics

Master Thesis

A comparison study of prediction approaches for multiple training data sets and test data with block-wise missing values

Author:
Frederik
Ludwigs

Supervisor:
Dr. Roman
Hornung

May 6, 2020

Abstract

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

On October 1, 1990 the international scientific research project named *Human Genome Project* was launched, with the aim to sequence the first complete human genome ever [1]. After investments of totally \$2.7 billion and 13 years of research the sequencing was officially finished in 2003 [2]. Since then, on the one hand, there have been biomedical advances that have led to the identification of disease genes which in turn have led "to improved diagnosis and novel approaches in therapy" [[3], p. 14]. On the other hand there has been an "extraordinary progress [...] in genome sequencing technologies" [[4], p. 333] leading to a sharp drop in sequencing prices. Nowadays whole genome sequencing is available and affordable for everyone - e.g. 'Veritas Genomics' offers whole genome sequencing for ~\$700 [5].

Besides the 'genome' that carries the whole genetic material of an organism, there are also other types of '-omes', such as 'epigenomes', 'transcriptomes', 'proteomes' and 'microbiomes'. The time and costs to collect data from these different types of '-omes' have been reduced drastically ever since the completion of the Human Genome Project [[6], [7], [8], [9], [10], [11]]. The methods for "fast, automated analyses of large numbers of substances including DNA, RNA, proteins, and other types of molecules" [12] are summarized under the term 'High Throughput Technologies'. These technologies make data from molecular processes available for many patients on a large scale.

The collected data from any type of '-omes' is commonly referred to as 'omics data'. In the clinical context it is of utmost interest to incorporate such omics data into different statistical approaches. A common example in this context is the survival time prediction for cancer patients, where in addition to regular clinical data gene expression data has been incorporated into the survival models. This additional omics data has "often been found to be useful for predicting [the] survival response" [[7], p. 1]. In "the beginning, only data from single omics was used to build such prediction models, together or without [...] clinical data" [[13], p. 1]. The usage of multiple distinct types of '-omes' in a single prediction approach was the next logical step and coined the term 'multi-omics data'. The theoretical aspects of integrating multiple omics types into a single prediction model and how to deal with the block-wise structures have been topic of several papers already - e.g. [13], [14], [15], [16], [17].

This thesis deals with a special type of missing data "that is common in practice, particular in the context of multi-omics data" [18] - the so called 'block-wise missingness'. Data with block-wise missingness consists of different folds and feature-blocks. While a feature-block stands for a collection of associated covariates, a fold represents a set of observations with the same observed feature-blocks. In data sets with block-wise missingness there is always at least one fold, with a missing feature-block, such that not all observations have the same observed feature-blocks.

Most statistical methods require fully observed data for training and predictions. In data with block-wise missingness this requirement is clearly not met, so that either the approaches need methodical adjustment or the data needs to be processed. This foundational problem with block-wise missingness raises the following challenges and questions: How can we fit a model on the block-wise missing data, without removing observations or whole feature-blocks? Does imputation work properly in these settings? How does a model that uses single feature-blocks only perform in comparison? How can a model predict on observations with missing feature-blocks?

In addition to the problem of block-wise missingness, there is also the challenge of "inherent high dimensionality" [[19] p. 93], when working with multi-omics data. Data from a single omics type can easily exceed thousands of covariates and the corresponding data sets usually consist of less observations than features [13]. Besides the predictive performance of an approach it is furthermore important for the approach to be sparse. "Sparsity is [...] an important aspect of the model which contributes to its practical utility" [[15], p. 3], as it makes the model much more interpretable than models including several thousands of variables.

A method that handles high dimensional data, even if the number of observations is lower than the amount of features, is the random forest method [13]. The method additionally handles different input types, does not need a lot of tuning and yields comparable predictive performances [20]. The only drawback is that it is not as interpretable as "models yielding [*in*] coefficient estimates of few relevant features" [[13], p. 35], as penalised regression approaches for example. Nevertheless variable importance measures can be extracted with the random forest method, as well as partial dependencies. Furthermore it has already been used successfully in various articles dealing with multi-omics data - e.g. [13], [14]. Moreover there have been proposals by Hornung et al. [18] and Krautenbacher [19] that modify the random forest approach, such that it can directly handle data with block-wise missingness. The different adaptations of penalised regression, as for example the IPF- & Priority-Lasso [[7], [15]] can also be modified so they can directly deal with block-wise missing data. The theoretical aspects of these approaches are not part of this thesis, but of Hagenberg's [21]. Nevertheless the performances of the different random forest approaches and penalised regression adaptations are also compared in this thesis.

Even though the problem of block-wise missingness is common in multi-omics data there are, to my knowledge, no comparison studies of such prediction approaches yet. Krautenbacher has already stated that “reliable analysis strategies for multi-omics data [...] [with block-wise missingness are] urgently needed” [[19] p. 94]. The thesis at hand aims to provide such a large scale comparison study of prediction approaches capable of dealing with block-wise missingness and shall help finding a reliable analysis strategy.

To investigate a reliable analysis strategy for multi-omics data with block-wise missingness this paper compares the predictive performance of two naive random forest approaches, a random forest approach on imputed data, two random forest adaptations and the adaptations of penalised regression. In the second chapter, firstly the term 'block-wise missingness' is defined in more detail and how it can arise in multi-omics data. Then a theoretical explanation of the random forest method for classification is given. Following three data processing approaches are explained - these process the block-wise missing data such that a regular random forest can be trained with it. Moreover two methodological adaptations of the random forest method are illustrated. These adaptations allow the random forest approach to directly deal with block-wise missing data. The first part of the third chapter covers general information to the used metrics and evaluation techniques. Following the different data sources and corresponding data sets are described and investigated. These data sets are then used to validate the performances of the various approaches. In the penultimate chapter all approaches are analysed and compared, while the last chapter of this thesis discusses all findings, draws a conclusion and gives an outlook.

2 Methods

This section deals with the theory of the random forest model and the different adaptions of it to handle data with block-wise missingness.
In the beginning block-wise missingness is defined in more detail and it is shown how it can arise in multi-omics data. Afterwards the theory of the random forest method for classification is illustrated. Subsequent three approaches that process the data with block-wise missingness, such that a regular random forest can be fit on them, are described. The last two sections of this chapter present two different adaptations of the random forest method. These adaptions enable the random forest method to directly deal with block-wise missing data.

2.1 Block-wise missingness in multi-omics data

Collecting omics data has become significantly cheaper and faster ever since the completion of the Human Genome Project. As a result, this type of data is used more and more frequently in the biomedical research - e.g. risk prediction of childhood asthma [19]. Even though the integration of multiple types of '-omes' into a single prediction approach seems promising there are still challenges to face. One of these challenges is a special type of missingness that is common in the context of multi-omics data, the so called block-wise missingness [18].
The term block-wise missingness needs to be defined in more detail, before clarifying how it can arise in multi-omics data. Table 1 shows a minimalist example for a data set with block-wise missingness, whereby the data consists of eight observations, 105 covariates and a binary response variable. While the covariates 'weight', 'height', 'income' and 'education' are pretty much self-explanatory, the features 'g_1', ..., 'g_{100}' could be any type of omics data. Data with block-wise missingness always consist of different blocks and folds. On the on hand, a **block** describes a set of covariates containing all features collected on the basis of a characteristic - basically all covariates that are related in content. The data in table 1 has three blocks in total. 'Block 1' consists of the variables 'weight' and 'height' representing the physical properties. 'Block 2' contains the variables 'income' and 'education' standing for economic properties. 'Block 3' includes the remaining variables 'g_1', ..., 'g_{100}' that are measurements from a single omics type and represent genetic properties. On the other hand, a **fold** represents a set of observations with the same observed feature-blocks - basically all observations with the same observed features. The data set in table 1 consists of three folds in total.

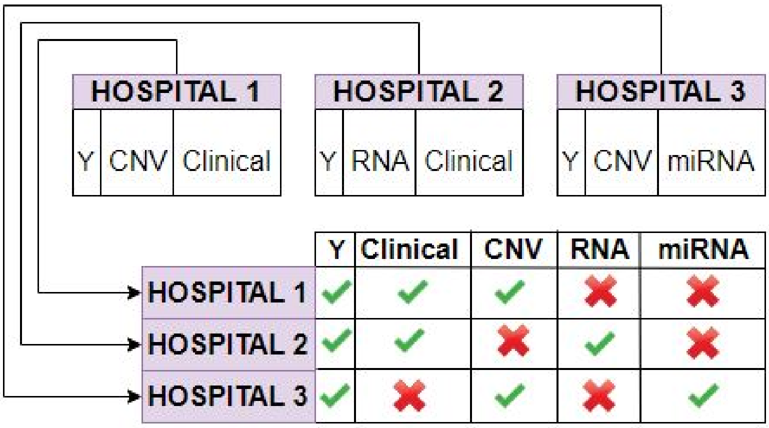
'Fold 1' holds the observations 1, 2 and 3, as these have the same observed feature-blocks ('Block 1' & 'Block 2'). 'Fold 2' holds the observations 4 and 5, while 'Fold 3' consists of the remaining observations 6, 7 and 8. As each fold has different observed feature-blocks, each fold is unique and every observation belongs to exactly one of them. The only variable all folds must have in common is the target variable.

ID	*weight*	*height*	*income*	*education*	g_1	$\cdots$	g_{100}	*Y*	
1	65.4	187	2.536	*Upper*				1	Fold1
2	83.9	192	1.342	*Lower*				0	
3	67.4	167	5.332	*Upper*				1	
4			743	*Lower*	-0.42	$\cdots$	1.43	1	Fold2
5			2.125	*Lower*	0.52	$\cdots$	-1.37	0	
6	105.2	175			-1.53	$\cdots$	2.01	0	Fold3
7	71.5	173			0.93	$\cdots$	0.53	0	
8	73.0	169			0.31	$\cdots$	-0.07	1	
	*Block*1		*Block*2		*Block*3				

Table 1: A data set with block-wise missingness - consisting of three blocks, three folds and the binary target variable 'Y'.

Multi-omics data with block-wise missingness have a structure as displayed in table 1, but the single feature-blocks are usually much higher dimensional than in the given example. When working with multi-omics data this type of missingness is a common problem. There are two main reasons for this:
The first one is related to the costs of collecting omics data. Even though the costs have been reduced drastically over the last 15 years, collecting omics data is still more complex and expensive than obtaining standard clinical data - e.g. 'weight', 'height', 'smoking status'. As a consequence, due to financial or even technical constraints, omics data can not always be collected for all participants of a study. Therefore participants from the same study can end up with different observed feature-blocks, such that the data for the whole study contains block-wise missingness.
The second reason is related to the collection of training sets from different sources - e.g. various hospitals. Even though the different sources do research regarding the same response variable - e.g. person has asthma - the surveyed feature-blocks can still differ. Therefore the concatenation of such data sets can result in a data set with block-wise missingness.
This scenario is illustrated in figure 1. In the top of the figure the three different data sources are displayed - 'Hospital 1', ..., 'Hospital 3'. Each

source consists of the target variable 'Y' and two feature-blocks as covariates - e.g. 'Hospital 2' consists of the target variable 'Y' and of the feature-blocks 'RNA' and 'Clinical'. The feature-blocks 'RNA', 'miRNA' and 'CNV' represent high dimensional omics data, while the 'Clinical' feature-block stands for several clinical features. Even though the target variable 'Y' is the same in all sources, the collected feature-blocks still differ. The concatenation of the data sets results in data with block-wise missingness and is displayed in the bottom of figure 1. In the concatenated data an observed block is marked with a green tick and a missing block with a red cross. The fold 'Hospital 2' only has 'RNA' and 'Clinical' as observed feature-blocks, such that the observations from this fold miss all the features from the blocks 'CNV' and 'miRNA'. The concatenated data totally consists of three unique folds and four different feature-blocks.



	Y	Clinical	CNV	RNA	miRNA
HOSPITAL 1	✓	✓	✓	✗	✗
HOSPITAL 2	✓	✓	✗	✓	✗
HOSPITAL 3	✓	✗	✓	✗	✓

Figure 1: Block-wise missingness, when concatenating data from diverse sources.

Training a prediction model is mostly not directly possible on data with block-wise missingness. Either the methods have to be adopted or the data processed. As block-wise missingness can also affect the test data it raises the following question. How can a model do a prediction for an observation that misses feature-blocks the model has originally been trained with? This challenge has to be taken into account when proposing methods capable to deal with block-wise missingness.

The remaining sections in this chapter focus on the approaches and adaptations to make model fitting on such data possible. Firstly the concept of the random forest for classification is explained and then the different approaches and adaptions to handle data with block-wise missingness.

2.2 Random Forest for classification

This chapter illustrates the random forest method that has already been applied in several articles dealing with multi-omics data [[13], [14], [15]]. It is a “powerful prediction method [...] able to capture complex dependency patterns between the outcome and the covariates” [[14] p. 2]. Furthermore it does not need a lot of tuning and naturally handles high-dimensional data, with more covariates than observations [13]. The random forest method can be applied to classification-, regression- and even survival-problems. Latter was added in 2008 by Ishwaran et al. [22]. As this thesis focuses on classification tasks, only the random forest for classification is explained. Nevertheless all of the approaches and adaptions described in sections 2.3 to 2.7 can also be applied for non-categorical target variables.
The random forest is a tree-based ensemble method that was introduced by Breiman in 2001 [23]. An ensemble is a concept from machine learning that “train[s] multiple models using the same learning algorithm” [24]. Therefore an ensemble consists of η identical so called base learners. The base learner of the random forest method is a ’decision tree’. This is an excellent base learner for an ensemble, as a decision tree can capture complex interactions and have a relatively low bias, if grown sufficiently deep. Especially as single decision trees are known to be noisy, they benefit from the ensemble [20]. Since decision trees are the basis of random forest method it is crucial to understand how these work in order to properly understand the random forest method.

2.2.1 Decision Tree

A decision tree is a supervised learning method that was introduced by Breiman et al. in 1984 [25]. It has a hierarchical nature, is easy to interpret and non-model based [26]. It applies recursive binary splitting to “partition the feature space into a set of rectangles” [[20] p. 305], such that the resulting rectangles are as pure as possible in terms of the target variable. A prediction is generated by assigning an observation to one of the rectangles in the partitioned feature space. The prediction then equals the distribution of the target variable within the assigned rectangle - e.g. a observation that falls into a rectangle with three negative and seven positive responses has predicted a probability of 70% for a positive response.
To partition the feature space into the purest rectangles possible the algorithm iterates over all possible split variable/ split value combinations. For each of these possible splits, the observations from the parent node N are divided - with respect to the split variable x_j at split point t - into the

child nodes N_1 and N_2 [[27], p. 10]:

$$N_1(x_j, t) = \{(x, y) \in N : x_j \geq t\} \tag{1}$$

$$N_2(x_j, t) = \{(x, y) \in N : x_j < t\} \tag{2}$$

Hence N_1 contains all observations from the parent node N with $x_j \geq t$, while N_2 contains all observations from the parent node N with $x_j < t$. The point $(x_j,\ t)$ therefore creates a binary split and partitions the data from parent node N in the two subspaces N_1 and N_2. The split variable x_j and split point t are chosen such that the resulting child nodes N_1 and N_2 have the greatest possible purity [27]. To measure the impurity of a node N regarding a categorical response with g classes the 'Gini-Index' (3), 'Misclassification-Error' (4) or 'Shannon-Entropy' (5) can be used [[27], p. 12]:

$$I(N) = \sum_{k=1}^{g} \hat{\pi}_{k,N} \cdot (1 - \hat{\pi}_{k,N}) \tag{3}$$

$$I(N) = 1 - \max_{k} \hat{\pi}_{k,N} \tag{4}$$

$$I(N) = -\sum_{k=1}^{g} \hat{\pi}_{k,N} \cdot \log(\hat{\pi}_{k,N}) \tag{5}$$

- $\hat{\pi}_{k,N}$: Relative frequency of category k in node N

For all of these impurity measures applies: The lower $I(N)$ the purer the node N and a node N is completely pure, when it only contains observations of the same response class - $I(N) = 0$. The corresponding plots of these impurity functions for a binary target variable are in the attachment in figure 17.

The reduction of the impurity when splitting the parent node N into the child nodes N_1 and N_2 is calculated by [[27], p. 10]:

$$I(N) - \frac{|N_1|}{|N|} \cdot I(N_1) - \frac{|N_2|}{|N|} \cdot I(N_2) \tag{6}$$

- $|N|$: Number of observations in the parent node N
- $|N_1|$: Number of observations in child node N_1
- $|N_2|$: Number of observations in child node N_2

This equation basically calculates how strong the impurity from the parent node N is reduced for a given splitting point that divides the observations to the child nodes N_1 and N_2. This impurity reduction is calculated for every possible split. The final split variable x_j and split point t are chosen, such that the impurity is maximally reduced.

For illustrative purposes the single partition steps of a classification tree are shown in figure 2. The figure consists of three plots in total, whereby each is a scatter plot of 'weight' and 'height' for the observations from 'Fold 1' and 'Fold 3' in table 1. Observations with a positive outcome are marked in blue, while negative outcomes are marked in red.

In the very beginning all observations are within the same feature space that has not been divided yet - the so called root node. This is displayed in the leftmost plot of figure 2. The root node - 'N1' - contains three observations with a positive and three with a negative response - hence the class distribution in this node is 50|50. The node is not pure regarding its responses and all possible impurity measures [(3), (4), (5)] have the highest possible value. The algorithm now iterates over all features, and for each feature over all possible split points and calculates the impurity of the resulting child nodes for each of these possible splits. The split variable and corresponding split value are chosen, such that the impurity reduction according to equation (6) is maximised. In the example of figure 2 the first split variable is chosen as 'weight' with the split value 69. Therefore the data from the root node - 'N1' - is split into the two child nodes 'N2' and 'N3' - central plot in figure 2. 'N2' contains the observations with a weight ≥ 69, while 'N3' consists of the observations with a weight < 69. The distribution of the target variable in 'N2' is 25|75 and in 'N3' 100|0. Hence both resulting child nodes are purer than their parent node 'N1'. The node 'N3' only contains observations with a positive response, therefore it is completely pure and can not be split any further - all possible impurity measures [(3), (4), (5)] have the lowest possible value. The node 'N2' on the other hand is not completely pure yet, and can be split further. 'N2' is now the parent node and the algorithm tries all possible splits on this segmented feature space. The highest impurity reduction of 'N2' is achieved with the split-variable 'height' on the value 171. 'N2' is therefore further split into 'N4' - all observations from 'N2' with a height ≥ 171 - and 'N5' - all observations from 'N2' with a height < 171. As well 'N4', as 'N5' are completely pure and the impurity of these nodes can not be reduced any further. The final partitioned feature space is displayed on the rightmost plot in figure 2. Based on this final partitioned feature space predictions can be done by assigning observations to one of the segments in the feature space. An observation with weight = 90 and height = 185 for example falls into the segment 'N4' and has a predicted class probability of 100% for response class 0.

In summary: The decision tree algorithm tries to split the feature space, such that the resulting child nodes maximally gain purity regarding the target variable. This is done with an exhaustive search, trying all possible split variables and corresponding split points, choosing the one that maximises the reduction of impurity.

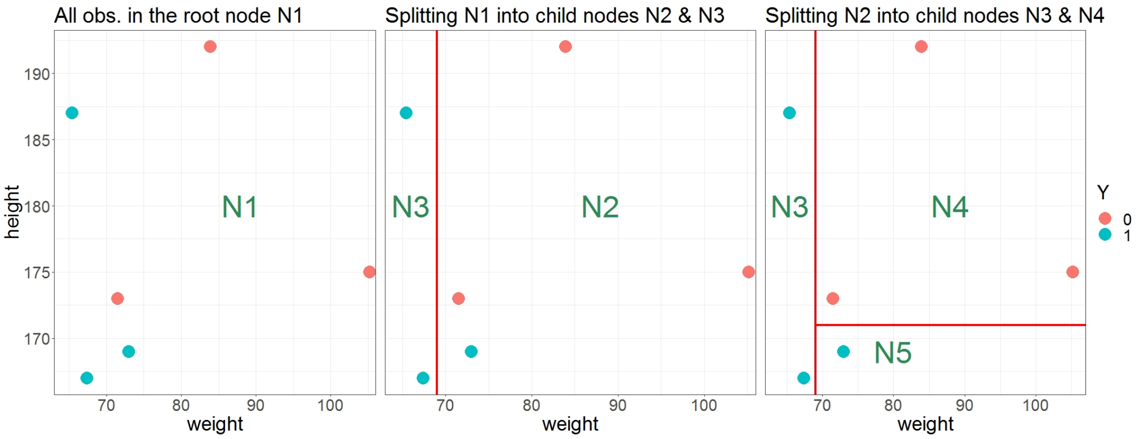


Figure 2: Recursive binary splitting of decision tree on a two-dimensional feature space.

A very handy property of a decision tree consists of its natural graphical display, which makes it extremely easy to interpret - even for people without mathematical background. This visualisation is especially useful, when the training data for the decision tree holds more than two covariates and can not be displayed as scatter plot [26].

The segmentation of the feature space from figure 2 is displayed as graphical decision tree in figure 3. Each square in the figure represents a node of the decision tree. Each of these nodes displays the response class with the highest proportion (top), the distribution of the response classes (mid) and the fraction of observations they contain (bottom). The split variables and split values are displayed below each node - nodes without a split variable/ value are ’terminal nodes’. The prediction for a test observation with figure 3 is very easy and intuitive. The test observation is simply passed down the decision tree until it reaches a terminal node. This is shown for a observation with weight = 90 and height = 185. The first node splits on the variable ’weight’ with the value 69. As the test observations has a weight $\geq$ 69 it is send down to the left child node. The next node splits on the variable height with the value 171. As the test observation is taller than 171cm it is send to the left child node. The observation is then in the node on the leftmost in the bottom of figure 3. This node is a terminal node and can not be divided further. The distribution of this node equals 100|0 and the prediction for the observations is therefore class 0 with 100% probability.

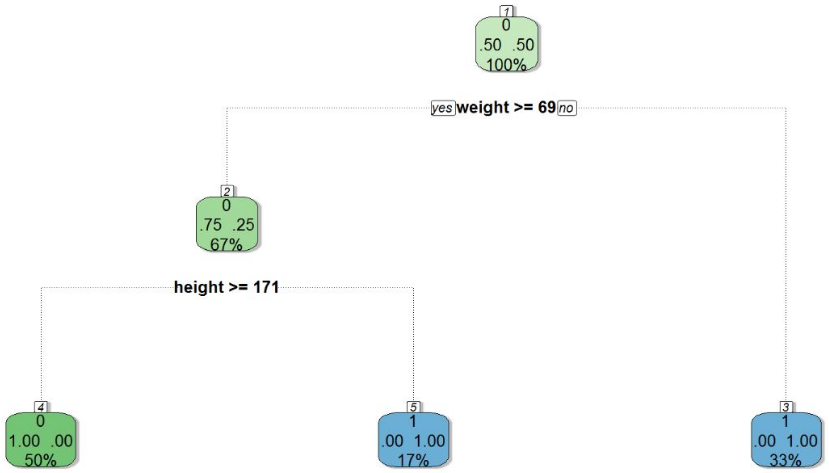


Figure 3: Corresponding decision tree for the segmented feature space on the rightmost plot in figure 2.

The complexity of a decision tree grows with the number of used splits and resulting terminal nodes [27]. The more complex a decision tree, the higher the chances of overfitting, while a tree with not enough complexity "might not capture the important structure[s]" [[26], p. 20]. So when should a tree stop with the binary partition of the feature space? There are multiple stopping criteria to control this, whereby the two most common used arguments are [26]:

- MinSplit:
  "The minimum number of observations that must exist in a node in order for a split to be attempted" [[28], p. 22]
- Complexity:
  "Split tree nodes only if the decrease in impurity due to splits exceeds some threshold" [[26], p. 20]

Both arguments have a huge impact on the complexity of a decision tree, as they control when the tree stops the partition of the feature space. The 'MinSplit' argument forces the tree to stop the partition, as soon as a potential parent node contains less than 'MinSplit' observations. The higher this argument is the earlier the tree has to stop growing and hence the less complex is the resulting tree. 'Complexity' on the other hand only allows splits that lead to a decrease of impurity of a given threshold when splitting the parent node N to N_1 and N_2. The drawback of this argument is that it is rather short-sighted, as a "seemingly worthless split might lead to a very good split below" [[26], p. 20]. Hence in this thesis the 'MinSplit' argument to control the complexity of a decision tree is preferred over the 'Complexity' argument.

The advantages of the decision tree method are numerous. It is easy to interpret, has no problems with outliers, captures interaction effects between features, handles categorical features and scales well with larger data [26]. Besides all these advantages, unfortunately there is also a huge disadvantage. A decision tree is highly unstable meaning that "small changes in the data could lead to completely different splits, thus, to a completely different tree" [[27], p. 26]. Even the removing of a single observation/ feature from the train data can lead to a completely different decision tree. The next chapter explains, how this alleged disadvantage of the high instability of a single decision tree is exploited by the random forest method to create predictions on the basis of multiple decision trees.

2.2.2 Random Forest Model

As already mentioned in the beginning of this chapter, the random forest is an ensemble method that uses the decision tree as a base learner. The random forest model therefore consists of multiple decision trees. To meaningfully train these multiple decision trees, the random forest method uses a modified version of bagging that was originally proposed by Breiman in 1996 [29]. As bagging is an important component of the random forest method it is explained in more detail.

To meaningfully train M base learners on a single data set, each base learner needs to be fit on a modified data set, else all the resulting learners are completely identical. To generate a different data set for each of the M base learners, bagging - short for **B**ootstrap **Agg**regation - is applied to the original data. In the first step bootstrapping is applied. It is a "type of resampling where large numbers of [...] samples of the same size are repeatedly drawn, with replacement, from a single original sample" [30]. The bootstrapping therefore generates M different bootstrap samples of the original data and trains any base learner B on these M bootstrapped data sets. To obtain a prediction from these, each of M fitted base learners is asked for a prediction - $B_m(x)$. These M different predictions are then aggregated for a final prediction: $B(x) = \frac{1}{M} \sum_{i=1}^{M} B_m(x)$ [[31], p. 4]. Bagging works best for learners with a high variance - e.g. a decision tree - as it reduces the variance of the base learner and increases only the bias in return [31].

The random forest method uses a slight modification of the bagging algorithm to construct bootstrapped decorrelated decision trees [20]. To do so the random forest algorithm does not only fit each decision tree on a separate bootstrapped data set, but decreases the correlation of these as well by randomly drawing 'mtry' features as possible split candidates at each split point instead of having all 'p' features as possible split candidates [31]. The standard value for 'mtry' with a categorical response class is $\lceil \sqrt{p} \rceil$, whereby p equals the number of covariates in the data. [32]. Hence at each node of a decision tree only a subset of the available features are drawn as possible split variables. Therefore every single decision tree has a different set of possible split variables for each their nodes. This modification of the original bagging algorithm therefore ensures that the trees are grown more diversely and the resulting trees are less correlated as with the regular bagging algorithm. The modified bagging algorithm to fit a random forest is the following [[20], p. 588]:

Algorithm 1: Growing a random forest model

```
Input  : D ← data of n observations & p features
         M ← number of trees in the forest
         $n_{min}$ ← 'MinSplit' argument of a decision tree
         mtry ← number of variables to draw at each split
for $m \leftarrow 1$ to $M$ do
    1. Draw a bootstrap sample Z* of size '$n$' from '$D$';
    2. Grow a decision tree of the random forest, by recursively
     repeating the following steps for each terminal node of the tree,
     until the minimum node size $n_{min}$ is reached;
       2.1 Randomly draw 'mtry' of the '$p$' available variables;
       2.2 Pick the best splitting point among the 'mtry' variables;
       2.3 Split the node into two daughter nodes;
```

The procedure to receive a prediction from a the random forest model is the same as in the original bagging algorithm. The input x is passed to each of the decision trees in the random forest and each of these trees creates a prediction based on the input x - details in section 2.2.1. The final prediction in case of a categorical response can either be the average of all M predicted class probabilities or the label that was predicted by the majority of the trees.

2.2.3 OOB error

A very handy property of the random forest method is the so called out-of-bag error (OOB error). The random forest model consists of multiple decision trees, whereby the data for each of these trees is obtained by drawing observations with replacement from the original data. For each tree, the average probability for an observation not to be drawn is ~ 0.37 [[31], p. 12]:

$$P(\text{Obs. not drawn}) = (1 - \frac{1}{n})^n \quad \xrightarrow{n \rightarrow \infty} \quad \frac{1}{e} \approx 0.37 \tag{7}$$

- n: Amount of observations in the data

Therefore each tree is grown with only $\sim 63\%$ of the available observations. The remaining $\sim 37\%$ of available observations are not used to grow the decision tree and can hence be used to get an estimate of the predictive performance of the random forest model - the so called OOB error. Before explaining the OOB error, lets have a look at figure 4. The figure displays the M different decision trees of a random forest model that was originally supplied with data of n observations and p features. Under each tree the data used for growing is displayed - a pink background indicates that the observation was 'in-bag' and hence used in the training of the decision tree. A grey background indicates that the observation is out-of-bag and was not used in the training of the decision tree. It should be noticed that the observations are drawn with replacement, so that an observation can enter the in-bag samples more than once.

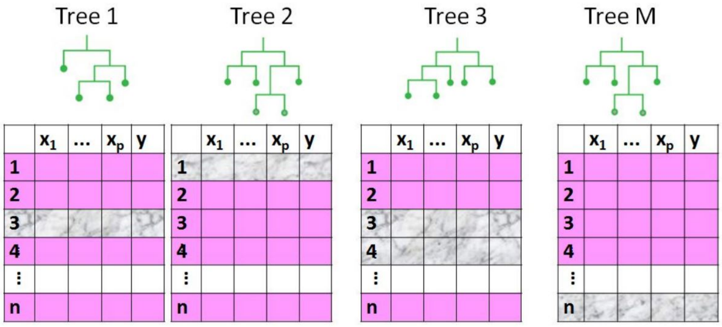


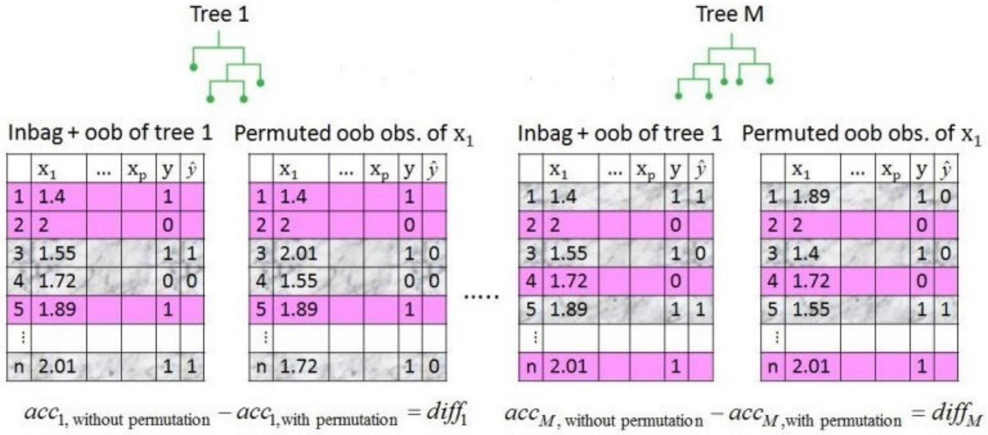
Figure 4: The data used to grow the M different decision trees of a random forest. Below each decision tree the in-bag observations are labelled in pink, while out-of-bag observations are labelled in grey [[31], p. 13].

To receive the OOB error of a random forest model, each of the M decision trees is asked for a prediction for the current observation i, but only if the observation i is an out-of-bag observation for the tree. This results in $\psi_i \leq M$ predictions for the observation i, whereby the final out-of-bag estimation for observation i equals the average of the ψ_i predictions. After receiving this out-of-bag prediction for all n observations, the final OOB error can be calculated. To compare the predicted classes and the true response class of the n observations any metric can be used - e.g. Accuracy, F-1-Score. The OOB error “is almost identical to that obtained by N-fold cross validation” [[20], p. 593]. Therefore, unlike most other prediction models, the random forest can be fit and evaluated in one single step - an extremely handy property.

2.2.4 Variable importance

In most applications, not all feature variables are equally important and mostly only a few have a relevant influence. Therefore the property of variable importance in the random forest method has a high practical usage. Even though the single decision trees of a random forest model are highly interpretable, the random forest model itself “lose[s] this important feature, and must therefore be interpreted in a different way” [[20], p. 367].
One possibility to measure the variable importance in a random forest model is based on the permutations of the out-of-bag observations. All out-of-bag observations of a decision tree T_m are passed to their tree for a prediction and the accuracy of the decision tree is recorded - $acc_{\text{m, without permutation}}$. To obtain the importance of a variable x_l, all out-of-bag observations of the decision tree T_m are permuted in the variable x_l, such that all out-of-bag observations receive a different value for the variable x_l. The permuted out-of-bag observations are then passed to the decision tree T_m and the accuracy of the decision tree is recorded again - $acc_{\text{m, with permutation in x}_l}$. The difference between the regular OOB accuracy - $acc_{\text{m, without permutation}}$ - and the OOB accuracy with permuted variable x_l - $acc_{\text{m, with permutation in x}_l}$ - is used as measure for the importance of the l-th variable in the decision tree T_m. The average importance of the l-th variable over all M decision trees equals the variable importance for x_l for the whole random forest model [31].
This technique to access the importance for the different variables is displayed in figure 5 for the variable x_1. For the decision trees 1 and M the data used to train these is displayed below. A grey background marks the out-of-bag observations. Based on these observations the out-of-bag accuracy can be calculated for each of the M trees. For each tree this results in $acc_{\text{m, without permutation}}$. Then the values of the variable x_1 are permuted for the out-of-bag observations for each decision tree. Following the out-of-bag accuracy is calculated with the permuted variable x_1 resulting in $acc_{\text{m, with permutation in x}_1}$. The difference diff_m between $acc_{\text{m, with permutation in x}_1}$ and $acc_{\text{m, without permutation}}$ represents the importance of variable x_1 in the decision tree T_m. The final importance of variable x_1 then equals the average

over all these differences [[31], p. 16]: $\frac{1}{M}\sum_{i=1}^{M} \text{diff}_i$



Inbag + oob of tree 1

	x_1	...	x_p	y	$\hat{y}$
1	1.4			1	
2	2			0	
3	1.55			1	1
4	1.72			0	0
5	1.89			1	
⋮					
n	2.01			1	1

Permuted oob obs. of x_1

	x_1	...	x_p	y	$\hat{y}$
1	1.4			1	
2	2			0	
3	2.01			1	0
4	1.55			0	0
5	1.89			1	
⋮					
n	1.72			1	0

$acc_{1,\text{ without permutation}} - acc_{1,\text{with permutation}} = diff_1$

Inbag + oob of tree 1

	x_1	...	x_p	y	$\hat{y}$
1	1.4			1	1
2	2			0	
3	1.55			1	0
4	1.72			0	
5	1.89			1	1
⋮					
n	2.01			1	

Permuted oob obs. of x_1

	x_1	...	x_p	y	$\hat{y}$
1	1.89			1	0
2	2			0	
3	1.4			1	0
4	1.72			0	
5	1.55			1	1
⋮					
n	2.01			1	

$acc_{M,\text{ without permutation}} - acc_{M,\text{with permutation}} = diff_M$

Figure 5: Calculation of the variable importance of x_1 for a random forest model consisting of M decision trees [[31], p. 16].

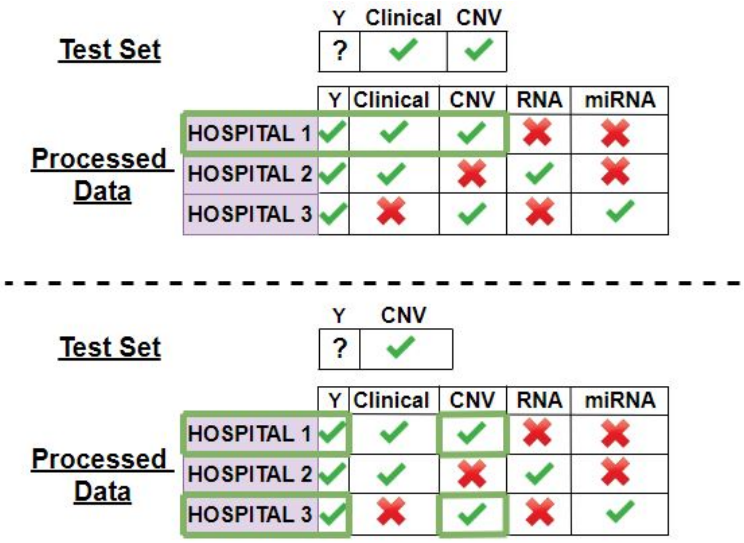
2.3 Complete-Case Approach

In this section the first baseline approach to handle data with block-wise missingness is explained - the so called 'Complete-Case' approach. This approach does not modify the random forest model itself, but processes the training data, such that it does not contain any missing values afterwards. This has the advantage that every prediction model - e.g. a random forest model - can be regularly trained on the processed data and the disadvantage of not using all available folds and feature-blocks. Therefore it is a rather simple approach and shall serve as a first baseline. The results from this first baseline approach are a hurdle to overcome for the more sophisticated approaches from the sections 2.5 - 2.7.

Let's have a look at the approach itself. As block-wise missingness can affect the test data as well as the training data it is possible that the test observations are missing feature-blocks - even if these are available in the training set. The 'Complete-Case' approach removes all folds from the training data that miss at least one of the available feature-blocks from the test data. The feature-blocks of the training data that are not available in test set are removed as well. After the processing, the training data only consists of the feature-blocks available in the test set with observations that were completely observed in these blocks. Based on this processed training set a random forest can be trained regularly. The prediction on the test observations with such a fitted model can then be done completely regular, as the model does not use any split variables that are not available for the test observations, as well as the test observations do not contain any features the model has not been trained with. To make the processing of the training data easier to understand two examples are shown in figure 6. In these examples the concatenated data with block-wise missingness from figure 1 is used as as a exemplary training data:

1. Example: This example is displayed in the top of figure 6. Even though the training data consists of four feature-blocks, the test observations have only two observed feature-blocks - 'Clinical' and 'CNV'. The 'Complete-Case' approach processes the training data, such that it removes all observations that miss at least one of the available feature-blocks of the test set. Therefore only observations from the fold 'Hospital 1' can be used, as the observations from the other folds either miss the feature-block 'Clinical' or 'CNV'. The fold and feature-blocks that can be used for the model fitting are marked with a green box. On this processed data a regular random forest model can be trained and used to create predictions for the test observations then. The processed training data contains of two feature blocks and two folds less than the original training data, as these were removed by the processing of the 'Complete-Case' approach.

2. Example: This example is displayed in the bottom of figure 6. The available training data originally consists of four feature blocks, while the test observations were only observed in the single feature-block 'CNV'. The 'Complete-Case' approach removes now all observations from the training data that do not have an observed 'CNV' feature-block. Therefore only the observations from the folds 'Hospital 1' and 'Hospital 3' can be used as training data. For these folds only the feature-block 'CNV' can be used for training and the other feature-blocks need to be discarded, as they are missing in the test set. The folds and feature-block that can be used for the model fitting are marked with a green box. On this data a regular random forest model can be trained and used to create predictions for the test observations then. As in the example above, much of the original training data is discarded by the 'Complete-Case' approach.



Test Set

Y	Clinical	CNV
?	✓	✓

Processed Data

	Y	Clinical	CNV	RNA	miRNA
HOSPITAL 1	✓	✓	✓	✗	✗
HOSPITAL 2	✓	✓	✗	✓	✗
HOSPITAL 3	✓	✗	✓	✗	✓

Test Set

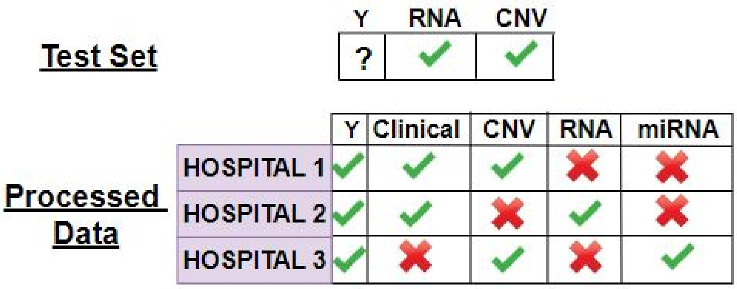
Y	CNV
?	✓

Processed Data

	Y	Clinical	CNV	RNA	miRNA
HOSPITAL 1	✓	✓	✓	✗	✗
HOSPITAL 2	✓	✓	✗	✓	✗
HOSPITAL 3	✓	✗	✓	✗	✓

Figure 6: The 'Complete Case' processing of the training data according to the available feature-blocks in the test set.

Besides the generous discarding of training data, the method has another big disadvantage. As the 'Complete-Case' approach removes all observations from the training set that miss at least one of the available feature-blocks of the test set, it may happen that there are no training observations left after the processing. This situation is displayed in figure 7. The test set in the figure contains the feature-blocks 'RNA' and 'miRNA' as observed feature-blocks. But as no fold in the training data was observed with both of these feature-blocks, the 'Complete Case' approach is not applicable, as the processing of the training data results in an empty data frame.



Test Set

Y	RNA	CNV
?	✓	✓

Processed Data

	Y	Clinical	CNV	RNA	miRNA
HOSPITAL 1	✓	✓	✓	✗	✗
HOSPITAL 2	✓	✓	✗	✓	✗
HOSPITAL 3	✓	✗	✓	✗	✓

Figure 7: The 'Complete Case' processing results in a empty training set, such that no model can be trained. In these settings, predictions can not be generated for the test set with the 'Complete Case' approach.

In summary, the 'Complete Case' approach removes all observations that miss at least one of the observed feature-blocks in the test set. Also all feature-blocks from the training data that are not available in the test set are removed. This data processing approach can discard a big part of the original training data and hence this approach does not handle the data very efficiently.

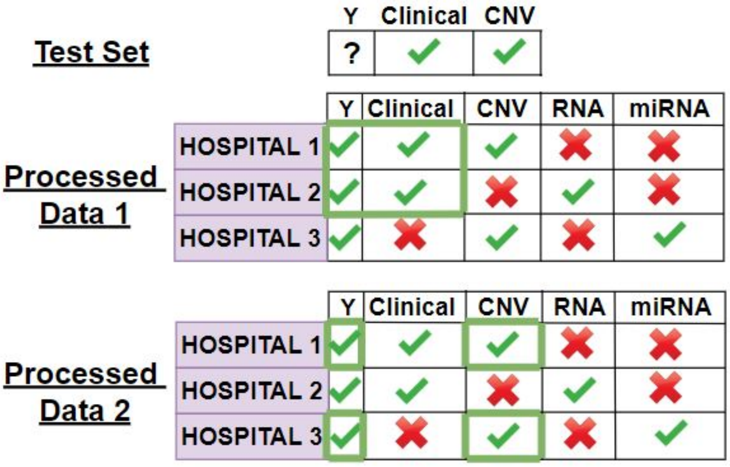
2.4 Single-Block Approach

The second baseline approach to handle data with block-wise missingness is the 'Single-Block' approach. As well as the 'Complete-Case' approach, it does not modify the random forest model itself but processes the training data, such that it does not contain any missing data anymore. As well as the 'Complete-Case' approach, the 'Single-Block' approach discards much of the available training data. As this approach is rather naive it is the second baseline approach and shall serve as another lower limit for the performances of the more sophisticated approaches from the coming sections 2.5 - 2.7.
As the name of the approach already suggests, it only uses a single feature-block to train a random forest model and predicts on the test set then. In order to create predictions on the test set the model must be trained with a feature-block that is also available in the test set. Else the fitted model can not predict on the test set, as it uses split variables that are not available in the test data. Hence the single feature-blocks from the training data that can be used to train a model depend on the observed feature-blocks in the test set. The concept of this approach is now explained with the example in figure 8. The training data in this example has already been introduced in the section 2.1 and has been used as an example in the previous section as well:
Example: The test set for this example is displayed in top of figure 8 and contains two different feature blocks - 'Clinical' and 'CNV'. The training data consists of four different feature blocks and three different folds in total. The 'Single-Block' approach processes the training data in multiple ways to get rid of the block-wise missingness in the training set. For each available feature-block in the test set it is checked, whether the training data involves the feature-block as well. For each feature-block that the test and training set have in common a separate random forest model is fitted and used to predict the outcome of the test observations. In the current example it is firstly checked, whether the training data involves a 'Clinical' or a 'CNV' feature-block. In this example the training data involves both feature-blocks of the test set. For each of the feature-blocks the test and training set have in common a separate processed data set is created.

Processed Data 1: As the test and training set have the feature-block 'Clinical' in common, the first processed training data only consists of the response Y and the feature-block 'Clinical' for the observations that have been observed in this block. This subset of the data is displayed in the middle of figure 8 and marked with a green box. Based on this subset, a random forest model can be regularly fit and used to create predictions for the test set. For the predictions on the test set only the features from the 'Clinical' feature-block can be used.

Processed Data 2: As the test and training set also have the feature-block 'CNV' in common, another processed training set is created. The processed training data then only consists of the response Y and the feature-block 'CNV' for the observations that were observed in the 'CNV' block. This subset of the data is displayed in the bottom of figure 8 and marked with a green box. Based on this data, a random forest model can be regularly fit and used to create predictions for the test set. For the predictions on the test set only the features from the 'CNV' feature-block can be used.

Predictions: As the processing of the training data with the 'Single-Block' approach results in two processed training sets, this approach consists of two different fitted models then. One random forest model that was fitted on the 'Clinical' feature-block and one random forest model that was fitted on the 'CNV' feature-block. Both of the fitted models can create predictions for the test set based on the features they have been trained with. The 'Single-Block' approach can therefore result in multiple predictions for the observations in the test set.



Test Set

Y	Clinical	CNV
?	✓	✓

Processed Data 1

	Y	Clinical	CNV	RNA	miRNA
HOSPITAL 1	✓	✓	✓	✗	✗
HOSPITAL 2	✓	✓	✗	✓	✗
HOSPITAL 3	✓	✗	✓	✗	✓

Processed Data 2

	Y	Clinical	CNV	RNA	miRNA
HOSPITAL 1	✓	✓	✓	✗	✗
HOSPITAL 2	✓	✓	✗	✓	✗
HOSPITAL 3	✓	✗	✓	✗	✓

Figure 8: 'Single-Block' processing of the training data so a random forest model can be regularly trained with each of these processed data sets.

In summary, the 'Single Block' approach creates an own processed training set for each of the feature-blocks the test and training set have in common. Each of the resulting processed training sets consist of only one single feature-block and do not contain any missing data, as the observations with missing values are removed. On each of these processed training sets a random forest model can be trained and used for predictions on the test set. As the 'Complete-Case' approach, the 'Single-Block' does not handle the data very efficiently.

2.5 Imputation Approach

This section introduces an approach that deals with block-wise missingness by imputing the missing values - the so called 'Imputation' approach. Other than the 'Complete Case' and 'Single-Block' approach, the 'Imputation' approach does not discard any of the available data and therefore uses the data more efficiently. After the missing values in the training data have been imputed, any prediction model can be fit on this data in a regular way and provide predictions for a test observation based on a single feature-block, or on the basis of multiple different feature-blocks.
"Many established [prediction] methods [...] require fully observed datasets without any missing values" [[33], p. 112] - in data with block-wise missingness this requirement is clearly not fulfilled. The idea to deal with this missingness by imputing the missing values is kind of obvious. The imputed data sets do not contain any missing values after the imputation, such that a prediction approach can be fitted regularly then. There are two big drawbacks when imputing missing values in multi-omics data. First, multi-omics data with block-wise missingness can consist of "many missing values making imputation techniques unreliable" [18]. Second, if the data is a concatenation of data sets from different sources, the imputation is "performed across different, potentially heterogeneous data sets" [18] - another reason for the unreliability of the imputation. Despite these disadvantages the 'Imputation' approach is still worth being compared to the other approaches in this study. Firstly a suitable imputation approach has to be found. This is not trivial, as multi-omics data usually do not only have less observations than features, but also a mixture of continuous and categorical variables as features [33]. Furthermore "such datasets often contain complex interactions and non-linear relation structures which are notoriously hard to capture" [[33], p. 112]. The 'k nearest neighbours' imputation method [34] requires at least one complete case observation, an assumptions that is not always true for multi-omics data with block-wise missingness. The 'Amelia' imputation [35] "assumes the data is distributed multivariate normal" [[36], p. 7]. Most

variables in multi-omics data do not fulfil this assumption and need a transformation to fulfil it - a not very handy property with such high-dimensional data [36]. There are more imputation methods, but most of these “are restricted to one type of variable” [[33], p. 112] or “make assumptions about the distribution of the data” [[33], p. 112]. An imputation method that can handle any type of input data and makes as few as possible assumptions about the data is based on the random forest method - the so called ’Miss’Forest’ [33]. This imputation approach needs “no tuning parameter, and hence it is easy to use and needs no prior knowledge about the data” [[33], p. 113]. Additionally it was shown that the ’MissForest’ approach is competitive to the ’k nearest neighbours’ and ’MICE’ imputation [33]. Furthermore the ’MissForest’ imputation can handle “mixed-type data and is known to perform very well under barren conditions like high dimensions, complex interactions and non-linear data structures” [[33], p. 113]. Because of all these mentioned advantages the ’MissForest’ method is used as imputation method in this thesis. The imputation approach is explained in more detail now.

will be explained in the following?

MissForest: The ’MissForest’ imputation method was proposed by Stekhoven and Bühlmann in 2012 [33] and builds up on the random forest method. For the imputation of missing values a random forest is trained on the observed parts of the data and then used to predict the missing values in the data. For the explanation, assume D to be a $n \times p$ dimensional data set with missing values in the diverse variables. For the imputation of a variable X_j with missing values at the entries $i^{(j)}_{mis} \subseteq \{1, \dots, n\}$ the data set D is separated into four parts [33]:

1 y^j_{obs}: Observed values of variable X_j

2 y^j_{mis}: Missing values of variable X_j

3 x^j_{obs}: Variables other than X_j with observations $i^{(j)}_{obs} = \{1, \dots, n\} \backslash i^{(j)}_{mis}$
typically not fully observed as the index $i^{(j)}_{obs}$
corresponds to the observed values in in X_j

4 x^j_{mis}: Variables other than X_j with observations in $i^{(j)}_{mis}$
typically not completely missing as the index $i^{(j)}_{obs}$
corresponds to the observed values in in X_j

The imputation procedure is explained and shown in algorithm 2 - algorithm and explanation are based on [33]:
In the beginning all missing values in the data set D are imputed with an

initial guess - e.g. mean imputation. In the next step the variables with missing values are ordered according to the amount of missing values - starting with the variable with the fewest missing values. For each of these variables a random forest model is fitted with the response y^j_{obs} and x^j_{obs} as predictor variables. With this fitted random forest model the missing values y^j_{mis} are imputed by the predictions of the random forest model based on x^j_{mis}. This procedure is repeated for a fixed amount of iterations or until the stopping criterion γ is met.

accordingly

Algorithm 2: Imputation procedure of the 'MissForest'

```
Input  : D ← data of n observations & p features
         $\gamma$ ← stopping criterion
1. Make initial guess for missing values - e.g. mean imputation;
2. k ← vector of sorted indices of the variables in $D$ w.r.t.
   increasing amount of missing values;
while not $\gamma$ do
    1. $D^{\text{IMP}}_{old}$ ← store previously imputed data;
    for $j \in \boldsymbol{k}$ do
        1. Fit a random forest: $y^s_{obs} \sim x^s_{obs}$;
        2. Predict: $y^s_{mis}$ using $x^s_{obs}$;
        3. $D^{\text{IMP}}_{new}$ ← update imputed matrix, using predictions $y^s_{mis}$;
    2. Update $\gamma$;
3. Return the imputed data set $D^{\text{IMP}}$;
```

The stopping criterion γ measures the difference between the newly imputed data matrix D^{IMP}_{new} and the previous imputed data D^{IMP}_{old}. For continuous variables $\mathbf{N}$ the difference is calculated via [[33], p. 113]:

$$\triangle_{\mathbf{N}} = \frac{\sum_{j \in \mathbf{N}} (D^{\text{IMP}}_{new} - D^{\text{IMP}}_{old})^2}{\sum_{j \in \mathbf{N}} (D^{\text{IMP}}_{new})^2} \tag{8}$$

And for categorical variables $\mathbf{F}$ it is calculated via [[33], p. 113]:

$$\triangle_{\mathbf{F}} = \frac{\sum_{j \in \mathbf{F}} \sum_{i=1}^{n} \mathbb{1}_{D^{\text{IMP}}_{new} \neq D^{\text{IMP}}_{old}}}{\#\text{NA}} \tag{9}$$

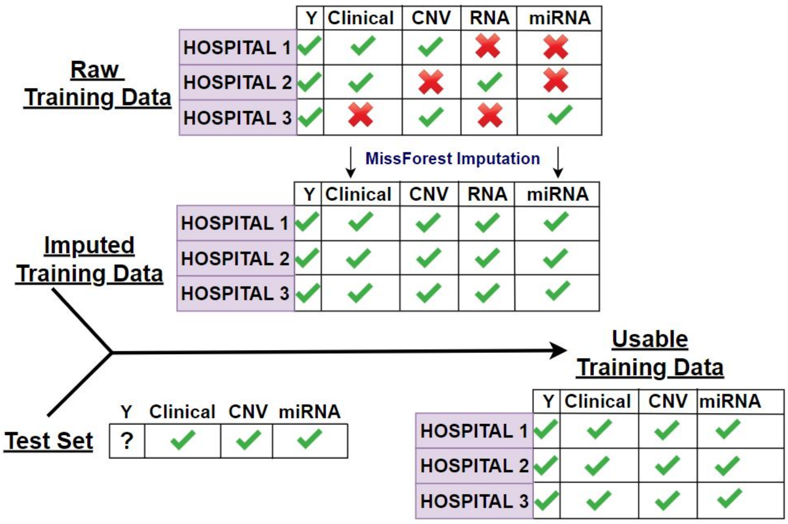
- #NA: Number of missing values in the categorical variables

The stopping criterion γ is fulfilled as soon as “the difference between the newly imputed data [...] and the previous one increases for the first time with respect to both variables” [[33], p. 113]. An alternative to the stopping criterion γ is a fixed amount of iterations for the imputation.

Predictions: Now that is has been clarified how the ’MissForest’ imputation works, the procedure of the ’Imputation’ approach can be explained in more detail based on the example in figure 9. The training data has already been introduced in the section 2.1 and was used as example in the previous sections aswell.

In the top of the figure the training-data with block-wise missingness is displayed. The very first step of the ’Imputation’ approach is to impute the missing values in the training-data with the ’MissForest’ method. After the imputation has taken place, the training-data does not contain missing values any more and is displayed right below the original training data. As this data has no missing values at all, a random forest model can be fit completely regular. But as the test-set might miss feature-blocks, the random forest model is only trained with the feature-blocks that the training- and test-set have in common. Else it might be possible that the fitted random forest model can not create predictions for the test-set, as it uses split variables that are not available for the test-set. Therefore all feature-blocks that are not available for the test-set have to be removed. Hence the imputed training-data that can actually be used to train a random forest model then only consists of the feature-blocks that are also in the test-set. Based on this usable training-data a random forest model can be fit in a regular way and can then provide predictions for the test-set.

regularly



	Y	Clinical	CNV	RNA	miRNA
HOSPITAL 1	✓	✓	✓	✗	✗
HOSPITAL 2	✓	✓	✗	✓	✗
HOSPITAL 3	✓	✗	✓	✗	✓

	Y	Clinical	CNV	RNA	miRNA
HOSPITAL 1	✓	✓	✓	✓	✓
HOSPITAL 2	✓	✓	✓	✓	✓
HOSPITAL 3	✓	✓	✓	✓	✓

Y	Clinical	CNV	miRNA
?	✓	✓	✓

	Y	Clinical	CNV	miRNA
HOSPITAL 1	✓	✓	✓	✓
HOSPITAL 2	✓	✓	✓	✓
HOSPITAL 3	✓	✓	✓	✓

Figure 9: Workflow of the ’Imputation’ approach to deal with block-wise missingness

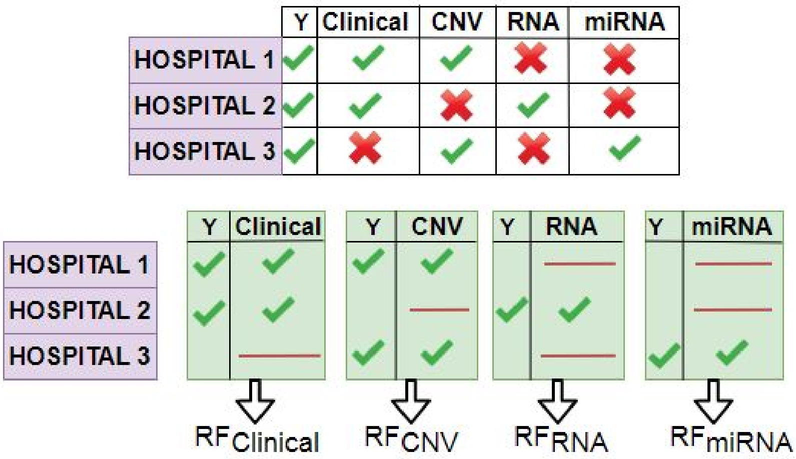
after this/that/this step... the imputation..

Summary: With the 'Imputation' approach, the missing values in the training-data with block-wise missingness are imputed by the 'MissForest' method. After the imputation training-data does not consist of missing values anymore. Based on the feature-blocks ~~that~~ the training-set and test-set have in common a regular random forest model can be trained. This fitted random forest model can then provide predictions for the test-set.

2.6 Block-wise Approach

This section introduces the 'block-wise' approach that was originally proposed by Krautenbacher in 2018 [19]. Other than the approaches from the previous sections 2.3 - 2.5 this approach does not modify the training data, but the random forest model itself. The 'block-wise' approach can directly handle block-wise missingness in the training data and does not need to process the data at all. Therefore it uses the available training data efficiently and does not discard any observations or feature-blocks. Furthermore a 'block-wise' fitted random forest is flexibly applicable and can provide predictions for a test observation based on a single feature-block, but also for a test observation on the basis of multiple different feature-blocks.
As the name of the approach already suggests, the random forest model is fitted in a 'block-wise' manner to the training data. In the beginning all available feature-blocks of the training data are extracted. On each of these feature-blocks a random forest model is separately fitted. This enables “all observations per [feature-block] [...] to be utilised for learning” [[19], p. 102] and no observation or feature-block has to be left out. With the 'block-wise' approach as many separate random forest models are fitted as the training data has feature-blocks. To create a prediction for a test observation then, each block-wise fitted random forest model is asked for a prediction. The models that were fitted on a feature-block that is not available for the test observation can not create a prediction, as these use split variables that are not available for the test observation. The remaining random forest models can create a prediction for the test observation by using the features from the test observation the models have originally been trained with. The predictions from the separate block-wise fitted models can then be aggregated to obtain a final prediction. The separate model fitting is explained in more detail with the example in figure 10. The training data in this example has already been introduced in the section 2.1 and has been used as example in the previous sections as well:
Model Fitting: The training data is displayed in the top of figure 10 and consists of four feature-blocks and three folds. To fit a separate random forest model on each feature-block, the training data needs to be split, such

that each feature-block can be used to train a random forest model. This is done by merging each feature-block and the response Y to a separate training set. In figure 10 these separate training sets are displayed as data frames with a green background below the original training data. From each of these separate training sets, all folds that contain missing values in the corresponding feature-block are removed - e.g. in the separate 'Clinical' training set all observations from 'Hospital 3' had to be removed, as this hospital did not collect any clinical data. The folds that had to be removed from these separate training sets are marked with a red horizontal line, while the usable folds are marked with a green tick. Based on each of these four different training sets a random forest model can be trained. This results in four distinct random forest models in total - $RF_{Clinical}$, RF_{CNV}, RF_{RNA} and RF_{miRNA}. Each of these models has been trained with a single feature-block only - e.g. RF_{RNA} was only trained with the feature-block 'RNA'.



	Y	Clinical	CNV	RNA	miRNA
HOSPITAL 1	✓	✓	✓	✗	✗
HOSPITAL 2	✓	✓	✗	✓	✗
HOSPITAL 3	✓	✗	✓	✗	✓

	Y	Clinical	Y	CNV	Y	RNA	Y	miRNA
HOSPITAL 1	✓	✓	✓	✓		—		—
HOSPITAL 2	✓	✓		—	✓	✓		—
HOSPITAL 3		—	✓	✓		—	✓	✓

Figure 10: Training of random forest models with the 'block-wise' approach.

The 'block-wise' approach trains the separate random forest models on the distinct feature-blocks in the training data and has as many separate random forest models as the training data consists of distinct feature-blocks. But how can these models be used to create a final prediction? As already mentioned, the block-wise predictions from the different random forest models need to be aggregated for a final prediction. This is explained in the following paragraph based on the example in figure 11.

Predictions: Assume that the four block-wise fitted random forest models from figure 10 can be used for the example in figure 11. The test set is displayed at the top of figure 11. For the observations in this test set the outcome 'Y' needs to be predicted. Other than the training data from figure 10, the test set only contains three feature-blocks and misses the 'CNV' feature-block from the training data. To create predictions for the observations in the test set, each of the four block-wise fitted random forest models is asked to create predictions for the test observations. As the test data contains the feature-blocks 'Clinical', 'RNA' and 'miRNA' the corresponding random forest models $RF_{Clinical}$, RF_{RNA} and RF_{miRNA} can be used to predict on the test data. The random forest model RF_{CNV} can not create predictions on this test set, as the feature-block 'CNV' is not available. Each of the three block-wise fitted models $RF_{Clinical}$, RF_{RNA} and RF_{miRNA} create a prediction for each observation in the test set, by only using the variables from the feature-block the models have originally been trained with. Therefore each model creates a prediction for each observation in the test set, such that there are three predicted outcomes for each observation - Preds$_{Clinical}$, Preds$_{RNA}$ and Preds$_{miRNA}$. These predictions represent the probabilities for each of the possible response classes. The final predictions for the target variable 'Y' equals a weighted average of these predictions.

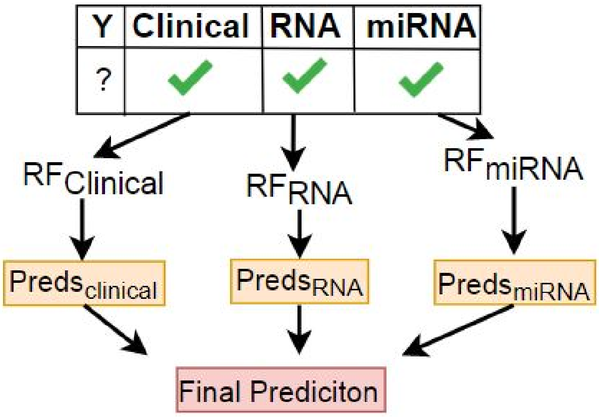


Figure 11: Prediction on test data with the 'block-wise' approach. The fitting of the random forest models was described with figure 10.

To create a meaningful weighted average of the different block-wise predictions, different techniques can be applied. The simplest method is giving each block-wise fitted model the same weight and return the plain average over all block-wise predictions. But as the block-wise fitted models have been trained on different feature-blocks this might not always be optimal, as the models might differ strongly in their prediction quality. To make this clear let us assume that the feature-block 'miRNA' is not related at all to the outcome 'Y', while the feature-block 'Clinical' is strongly related to it. In this case it can be assumed that the predictions based on the 'miRNA' feature-block are worse than the predictions based on the 'Clinical' feature-block. Therefore it would be meaningful to put a higher weight on the predictions from the $RF_{Clinical}$ model than on the predictions from the RF_{miRNA} model. Usually the true strength of the relation between a feature-block and the target variable is unknown. Therefore the predictive quality of the different feature-blocks needs to be estimated. This can be done with the out-of-bag error of the block-wise fitted models. For each block-wise fitted random forest model the predicted classes for all out-of-bag observations are generated - see chapter 2.2.3 for details. Based on the predicted outcomes and the true response values any metric can be calculated to judge the predictive quality of a block-wise fitted random forest model then. In this thesis either the accuracy or F-1-Score is used as metric to judge the predictive quality - details to the metrics in chapter 3.1.1. The better the out-of-bag accuracy/ F-1-Score of a block-wise fitted model, the higher the estimated predictive quality of the model. The higher the predictive quality of a model, the higher its weight and therefore the higher its contribution to the final prediction. The reason to use the F-1-Score besides the accuracy is that the F-1-Score is sensitive to class imbalances in the target variable, while the accuracy only represents the fraction of correctly classified observations.

Let us have a look at a minimalist example to make the idea of the weighted average clearer. Assume the block-wise fitted random forest models from figure 11 have the following out-of-bag accuracy and predicted probabilities for a positive response class for the observation i:

$OOB_{\text{Acc}}(RF_{Clinical}) = 0.67$ $\quad$ $\text{Preds}_{Clinical}(\text{Obs}_i) = 0.19$

$OOB_{\text{Acc}}(RF_{RNA}) = 0.86$ $\quad$ $\text{Preds}_{RNA}(\text{Obs}_i) = 0.33$

$OOB_{\text{Acc}}(RF_{miRNA}) = 0.21$ $\quad$ $\text{Preds}_{miRNA}(\text{Obs}_i) = 0.99$

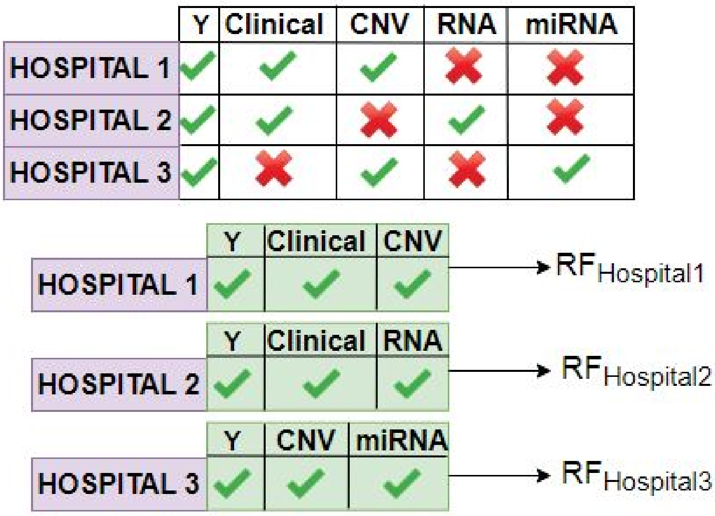
The predictions of the models represent the probability for a positive response, such that all probabilities < 0.5 result in a predicted negative class, while probabilities ≥ 0.5 result in a positive predicted response class. The true response class for the observation i is negative, as well as the predictions of $RF_{Clinical}$ and RF_{RNA}. Only the RF_{miRNA} model predicts the response for observation i wrongly as positive. When calculating the plain average of all these predicted probabilities it results in 0.503. Therefore the final predicted probability is ≥ 0.5 and the predicted class is the positive class - which is wrong for observation i. If we use the out-of-bag accuracy of the models as weights for a weighted average the the final predicted probability is $0.355 < 0.5$ and therefore the predicted class is the negative class - which is correct for observation i. So instead of giving all block-wise predictions the same weight, the predictive power of the single feature-blocks can be estimated with any out-of-bag metric and used to weight the predictions. This results in a higher weight for the predictions from models with a better out-of-bag metric.

In summary: With the 'block-wise' approach a separate random forest model is fitted on each feature-block of the training data. For a prediction on a test observation, all block-wise fitted models are asked for a prediction. Only those models that have been trained with a feature-block that is available for the test observation can create a prediction. These predictions can then be averaged in a weighted/ unweighted way to create a final prediction for the test observation.

2.7 Fold-wise Adaption

This section introduces the 'fold-wise' approach that was originally proposed by Hornung et al. [18]. This approach was originally not proposed to deal with block-wise missingness in multi-omics data, but to deal with multiple training set with the same target variable and different partly overlapping feature-blocks. Nevertheless, this approach can also deal with block-wise missingness in multi-omics data. Other than the approaches from the sections 2.3 - 2.5 this approach does not modify the training data, but the random forest model itself. The 'fold-wise' approach can directly handle block-wise missingness in the training data and does not need to process the data at all. Therefore it does not discard any of the available observations or feature-blocks and uses the available training data efficiently. Furthermore a 'fold-wise' fitted random forest is flexibly applicable and can "provide predictions for test data that do not feature all covariates available from training" [18]. As the name of the approach already suggests, the random forest model is fitted in a 'fold-wise' manner to the training data. In the beginning all available folds of the training data are extracted. On each of these folds a random forest model is then separately fitted. This results in as many fold-wise fitted random forest models as the training data has folds. As the different folds of the training data usually consist of multiple feature-blocks, each fold-wise fitted random forest model generally incorporates the covariates from multiple different feature-blocks. To receive a prediction on a test data only "the subsets of covariates included in the test data that are also included in at least one of the training data sets" [18] are used. The prediction of a single fold-wise fitted random forest model is then obtained as follows: (1) Remove all trees from the fold-wise fitted random forest that use a split variable as first split that is not available for the test observation. These trees can not even split the test data once, as the first split variable is not available for the test observation. Therefore these decision trees are of no value for the given test observation. (2) For each remaining decision tree "follow each branch of the tree and cut the branch as soon as a covariate is used for splitting that is not available" [18] for the test data. This process of cutting branches is called 'pruning'. A node that had to be pruned is a new terminal node of the decision tree then. After these two steps have been applied to the fold-wise fitted model the predictions can be obtained as for a standard random forest model. The predictions from the separate fold-wise fitted models can then be aggregated to obtain a final prediction. The fold-wise model fitting is explained in more detail with the example in figure 12. The training data in this example has already been introduced in the section 2.1 and has been used as example in the previous sections as well:

Model Fitting: The training data is displayed at the top of figure 12 and consists of four feature-blocks and three folds. To fit a separate random forest model on each fold, the training data needs to be split, such that each fold can be used to train a random forest model. This is done by merging the feature-blocks of a fold and the corresponding response Y to a separate training set. The feature-blocks that were not observed for a certain fold are removed from the fold-wise training data - e.g. the feature-blocks 'RNA' and 'MIRNA' were not observed for the fold 'Hospital 1' and had to be removed from the training data of the fold. In figure 12 these separate training sets are displayed as data frames with a green background below the original training data. Based on each of these three different training sets a random forest model can be trained. This results in three random forest models in total - $RF_{Hospital1}$, $RF_{Hospital2}$ and $RF_{Hospital3}$. Each of these models has only been trained with the observed feature-blocks of the different folds - e.g. $RF_{Hospital1}$ was trained with the feature-blocks 'Clinical' and 'CNV'.



	Y	Clinical	CNV	RNA	miRNA
HOSPITAL 1	✓	✓	✓	✗	✗
HOSPITAL 2	✓	✓	✗	✓	✗
HOSPITAL 3	✓	✗	✓	✗	✓

	Y	Clinical	CNV
HOSPITAL 1	✓	✓	✓

	Y	Clinical	RNA
HOSPITAL 2	✓	✓	✓

	Y	CNV	miRNA
HOSPITAL 3	✓	✓	✓

Figure 12: Training of random forest models with the 'fold-wise' approach.

The 'fold-wise' approach trains the separate random forest models on the distinct folds of the training data and consists of as many separate random forest models as the training data consists of unique folds. But how can these models be used to create a prediction? As already mentioned, the fold-wise predictions from the different random forest models need to be aggregated for a final prediction. To receive a prediction from a fold-wise fitted random forest model the single decision trees of such a model might be pruned. Before explaining the aggregation of the fold-wise predictions it is important to understand pruning process. It is explained in the following paragraph with the help of figure 13:

Pruning: Pruning actually describes a process applied to decision trees to avoid overfitting. But it can also be applied to decision trees, if these contain split variables that are not available for a test observation for which a prediction is asked for. The latter idea is used in this 'fold-wise' approach and explained in more detail with the help of figure 13. On the left of the figure, the original decision tree from figure 3 can be seen. It was grown on the basis of the two feature variables 'weight' and 'height'. To obtain a prediction the observation is simply passed down the tree until it reaches a terminal node. The predicted probabilities equal the distribution of the target variable in the terminal node. But how can this decision tree be used to predict on a observation with an unknown 'height'? To receive such a prediction the original decision tree has to be pruned. For this all nodes that split with the variable 'height' needs to be cut off. This is displayed on the right side of figure 13. The scissors indicate the pruning at the node that uses 'height' as split variable. This node is a terminal node then. The pruned tree has one terminal node less than the original decision tree and can create predictions for observations without a 'height' variable, as the pruned decision tree does not use this variable as split variable anymore.

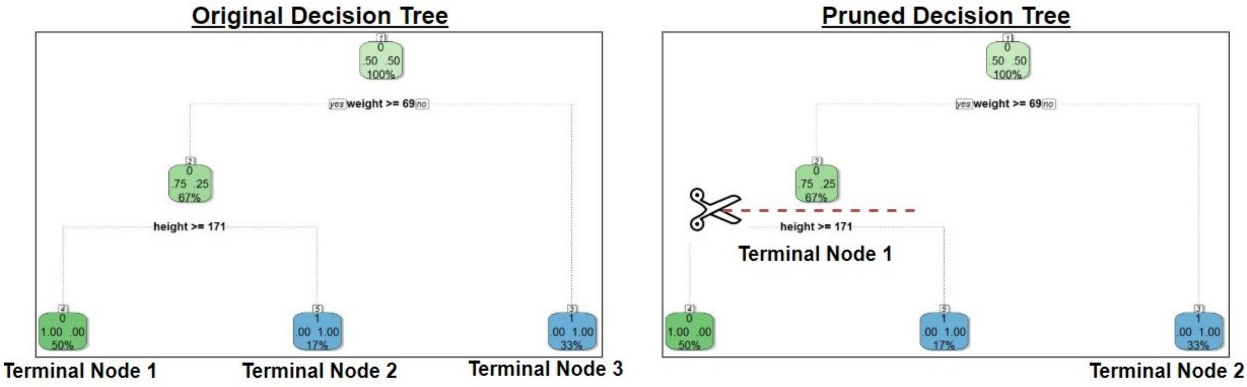


Figure 13: The pruning of a single decision tree. The decision tree was originally introduced in figure 3

The process of receiving a final prediction for an observation based on the predicted classes from multiple fold-wise fitted random forest models, is explained in the next paragraph based on the example in figure 14.

Predictions: Assume that the three fold-wise fitted random forest models from figure 12 can be used for the example in figure 14. The test set is displayed at the leftmost of figure 14, whereby the outcome 'Y' needs to be predicted. Other than the training data from figure 12, the test set only contains three feature-blocks and misses the 'CNV' block. To create predictions for the observations in the test set the three fold-wise fitted random forest models from figure 12 can be used. Each of the these models is asked for a prediction. As $RF_{Hospital2}$ was only trained on feature-blocks that are also available in the test set - 'Clinical' and 'RNA' - no tree needs to be pruned, as all of the used split variables in this random forest model are available for the test set. The prediction on the test set with $RF_{Hospital2}$ is therefore completely regular. The fold-wise fitted random forest models $RF_{Hospital1}$ and $RF_{Hospital3}$ were both trained under the inclusion of the feature-block 'CNV'. This feature-block is not available for the test observations from figure 14. Therefore the single decision trees in $RF_{Hospital1}$ and $RF_{Hospital3}$ need to be pruned, as these trees could contain nodes with split variables that are not available for the test observations. Firstly all decision trees of these models that use a 'CNV' covariate as first split variable have to be removed, as these trees can not even partition the test data once. Secondly all remaining trees are pruned as explained in the paragraph before. After applying these two steps to the models $RF_{Hospital1}$ and $RF_{Hospital3}$, the predictions for the test observations can be obtained “as in the case of a standard” [14] random forest model. In the end, each fold-wise fitted model creates a prediction for each observation in the test set, such that there are three predicted outcomes for each observation - Preds$_{Hospital1}$, Preds$_{Hospital2}$ and Preds$_{Hospital3}$. These predictions represent the probabilities for each of the possible response classes. The final predictions for the target variable ’Y’ equal a weighted average of these predictions then.

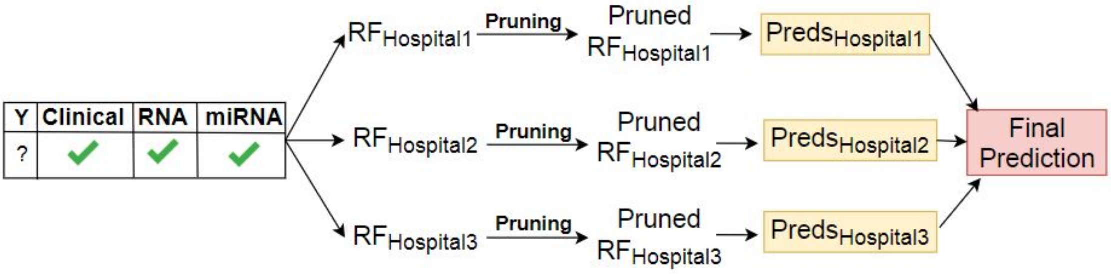


Figure 14: The prediction on test data with the ’fold-wise’ approach. The training of the models was described with figure 12.

To create a meaningful weighted average of the different fold-wise predictions, different techniques can be applied. The simplest method is giving each fold-wise fitted model the same weight and return the plain average over all the fold-wise predictions. This might not always be optimal, as the fold-wise fitted random forest models were trained with different combinations of feature-blocks and might differ in their predictive quality. As for the ’block-wise’ approach from section 2.6 the predictive quality of the different prediction models can be estimated with an out-of-bag metric - e.g. accuracy or F-1-Score. The out-of-bag metrics of the fold-wise fitted random forest models can then be used to weight the predictions of the different models. The higher the out-of-bag metric for a model, the higher the contribution of the model to the final prediction. There is only one difference in the out-of-bag calculation between the two approaches. As the fold-wise fitted random forest models might be pruned - depending on the test data. The out-of-bag predictions are generated with the pruned trees of the random forest then. This is meaningful, as pruning might reduce the predictive power of a model and we want to obtain a realistic estimation of the predictive power of a random forest model on the test set.

In summary: The fold-wise approach fits a separate random forest model on each fold of the training data. Depending on the available feature-blocks in the test set, the fold-wise fitted random forest models might be pruned. After the pruning process each fold-wise fitted model can generate predictions for the test set. These predictions can then be averaged in a weighted/ unweighted way to create a final prediction for the test observation.

3 Benchmark Experiments

This section deals with the benchmark experiment for the comparison of the predictive performance of different approaches on data with block-wise missingness. The comparison of the approaches is “necessary to ensure that [the] previously proposed methods work as expected in various situations” [[37], p. 3]. It is not expected that such experiments “result [in] an absolute truth applicable to all situations” [[37], p. 14f], but with enough different data sets and sufficient objective evaluation criteria general trends can be determined [37].

In the beginning the different metrics used in this thesis are introduced and defined. Afterwards it is explained how the predictive performance of an approach can be estimated if there is no separate test set. Subsequently the different data sources and corresponding data sets are investigated, as well as the diverse structures of block-wise missingness. Furthermore the evaluation technique for each data source is explained separately after the data sources have been introduced and investigated.

without the use of a
seperate test set?

3.1 Assessing the Performance

This section supplies the essential information needed to estimate the predictive performance of a classification model. In the first chapter the different metrics are introduced and defined. These are used to rate and compare the predictive performance of the different approaches. Subsequently ’k-fold cross validation’ is introduced and explained. This technique is used to estimate the predictive performance of a model when there is no separate test.

3.1.1 Metrics

This section introduces the diverse metrics used in this thesis to rate the predictive performances of the different approaches. Metrics are needed “in order to evaluate the performance of a statistical learning method on a given data set [...] [and] measure how well its predictions actually match the observed data” [[38], p. 29]. As this thesis aims to compare classification approaches, only performance metrics suitable for classification problems are introduced. The thesis only uses data sets with a binary response variable to compare the different approaches. Therefore the metrics are only explained for a binary response, but nevertheless the metrics can also be applied to multi-class problems. The selection of metrics is very important, as it influences how the performance of the approaches are measured and compared.

Confusion Matrix

The confusion matrix itself is not a performance measure, but the basis for most classification metrics [39]. It is used in classification problems where the response variable Y has at least two classes. The matrix always consists of as many rows and columns, as the response variable has unique classes.

can be seen in table2

An example of a confusion matrix is in table 2 and displays a matrix with the dimensions 2 × 2. The rows of the table represent the predicted class - $\hat{Y} \in [0;1]$ -, while the columns represent the actual class - $Y \in [0;1]$ [40]. Each cell of the matrix displays the amount of observations that were classified correctly | wrongly:

- **TP:** True Positives - amount of observations where the true outcome and the predicted outcome is both positive ($Y = 1$ & $\hat{Y} = 1$)
- **FN:** False Negatives - amount of observations where the true outcome is positive ($Y = 1$), but the predicted outcome is negative ($\hat{Y} = 0$)
- **FP:** False Positives - amount of observations where the true outcome is negative ($Y = 0$), but the predicted outcome is positive ($\hat{Y} = 1$)
- **TN:** True Negatives - amount of observations where the true outcome and the predicted outcome is both negative ($Y = 0$ & $\hat{Y} = 0$)

	Y = 1	Y = 0	
$\hat{Y} = 1$	**TP**	**FP**	Predicted Classes
$\hat{Y} = 0$	**FN**	**TN**	

True Classes

Table 2: Confusion matrix for a binary response.

Therefore **TP** and **TN** show the amount of observations that were labelled correctly, while **FN** and **FP** represent the wrongly labelled observations. Hence the confusion matrix shows the amount of correctly and wrongly labelled observations and can be used for the calculation of sophisticated classification metrics.

as a running..

The confusion matrix from table 2 is used as running example for the introduction of the different metrics in the next paragraphs.

Error-rate & Accuracy

“The most common approach for quantifying the accuracy [...] [of a prediction model is the] error-rate” [[38], p. 37]. It represents the proportion of mistakes a prediction model did when predicting the classes for the given set of observations - it can be calculated with help of the confusion matrix: **Error-rate** $= \frac{\mathbf{FP+FN}}{\mathbf{TP+TN+FN+FP}}$ [[40], p. 4].
The accuracy on the other hand represents the exact opposite, as it represents the fraction of correctly classified observations. It can also be calculated directly from the confusion-matrix: **Accuracy** $= \frac{\mathbf{TP+TN}}{\mathbf{TP+TN+FN+FP}}$ [[40], p. 4].
The error-rate/ accuracy are rather simple metrics and intuitively understandable. The accuracy is $\in [0, 1]$ and is better the higher its value, while the error-rate is $\in [0, 1]$ is worse the higher its value.
Both metrics are good measures when the classes of the response variable are nearly balanced, but should not be used if the classes of the target variable are highly imbalanced [39]. Think of a data set where only 1% of the observations is labelled as ’positive’ and the remaining 99% as ’negative’. If a model is really naive and always predicts a negative outcome, it still has a accuracy of 99% even though the model is terrible at predicting the actual outcome.
To overcome this drawback of the error-rate/ accuracy, other metrics are introduced in the following.

the.. accurracies are.. / the accuracy metrics are

gets better?

hinking of (ist weniger persönlich oder?)

Precision

The precision metric represents the fraction of observations with an predicted positive response - $\hat{Y} = 1$ - that also have an actual positive response - $Y = 1$. On basis of the confusion matrix it can be calculated via: **Precision** $= \frac{\mathbf{TP}}{\mathbf{TP+FP}}$ [[40], p. 4].
As the precision metric only respects the observations with a predicted positive class, it is less sensitive to class imbalances then the accuracy and error-rate metric. The precision metric is $\in [0, 1]$ and the lower its value the worse the predictive performance.

than (when=then hehe)

Recall

The recall metric represents the fraction of observations with an actual positive response - $Y = 1$ - that were correctly classified as positive - $\hat{Y} = 1$. It can be calculated on basis of the confusion matrix via: **Recall** $= \frac{\mathbf{TP}}{\mathbf{TP+FN}}$ [[41], p. 2].
As it measures how accurate the predictions for ~~the~~ observations with an actual positive outcome are, the recall metric ignores ~~the~~ observations with an actual negative outcome. Therefore this metric is less sensitive to class

zu viele Artikel

imbalances than the accuracy and error-rate metric. The recall metric is $\in [0, 1]$ and the higher its value the better the predictive performance.

F-1 Score

The F-1 score is a very common metric that represents the two metrics precision and recall at once. It is the harmonic mean of both metrics and based on the precision and recall it can be calculated via [[40], p. 4]:

$$\textbf{F-1 Score} = 2 * \frac{\textbf{Precision} * \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}} \tag{10}$$

The F-1 score is $\in [0, 1]$ and the better the predictive performance of a model the higher the F-1 score.

Balanced Accuracy

The balanced accuracy is a metric that "avoids inflated performance estimates on imbalanced data sets" [42]. Basically it calculates the accuracy for each possible response class separately and returns the average of these
accuracies class-wise accuracys. This average of the class-wise accuracys can be determined with the confusion matrix [43]:

$$\textbf{Balanced Accuracy} = \frac{\frac{\textbf{TP}}{\textbf{TP+FN}} + \frac{\textbf{TN}}{\textbf{TN+FP}}}{2} \tag{11}$$

As the regular accuracy it is $\in [0, 1]$ and the better the higher its value.

Matthews correlation coefficient

The Matthews correlation coefficient (MCC) is a metric that is insensitive to class imbalances. It is a "widely used performance measure in biomedical research" [[41], p. 1] and can be seen as a "discretization of the Pearson correlation for binary variables" [[41], p. 1]. It can directly be calculated with the confusion matrix via [[44], p. 415]:

$$\textbf{MMC} = \frac{\textbf{TP} * \textbf{TN} - \textbf{FP} * \textbf{FN}}{\sqrt{(\textbf{TP} + \textbf{FP}) * (\textbf{TP} + \textbf{FN}) * (\textbf{TN} + \textbf{FP}) * (\textbf{TN} + \textbf{FN})}} \tag{12}$$

? The MMC is in "essence a correlation coefficient" [42] with values $\in [-1, 1]$. 1 is the best possible value and represents perfect predictions, 0 indicates that the predictions are random on average and a value of -1 indicates the worst possible predictive performance [42].

3.1.2 k-fold Cross Validation

'K-fold cross validation' is a technique to estimate the predictive performance of a model, when there is no separate test set. Before explaining the technique in more detail, it is important to understand the difference between the calculation of a metric on the test- and training-set.

Regardless of the metric, there is a big difference whether the metric is calculated on the training- or on the test-set. The training-set is the data used to train a prediction model, while the test-set "was not used in training the model" [[38], p. 176]. The calculation of a metric on the training-set is often quite different from the metric obtained when calculating it on the test-set - "in particular the former can dramatically underestimate the latter" [[38], p. 176]. The calculation of a metric on the training data is therefore over-optimistic, as "the same data is being used to fit the method and assess its [predictive performance]" [[20], p. 228]. Hence the calculation of a metric should always be done with data that has not been used to train the model. In case of having no designated test set, there are diverse techniques to estimate the predictive performance by using only the available training data.

..consists
of the data
used..?

A very simple approach to obtain a metric on a test-set is the so called 'hold-out' method. With this technique a data set D is split to a training-set D_{train} and a test-set D_{test}. A model can then be trained on D_{train} and evaluated on D_{test} to see how well that model performs on unseen data. Such a single train-test split can be problematic and lead to distorted estimates, as the metric highly depends on how the data is split into training- and test-set. Different train-test splits of the data can therefore lead completely distinct results. In general, the smaller the test-set D_{test} "the higher the variance of our estimated metric" [[45], p. 18]. And the smaller the train-set D_{train}, the bigger the introduced pessimistic bias, as the model is trained on less data and will therefore learn less and perform worse [45].

To avoid this bias-variance trade-off with the 'hold-out' method, there are techniques that use the data more efficiently through resampling. Resampling in general describes the process of repeatedly splitting the training data D into training- and tests-sets, whereby the resulting metrics can be calculated for each of these splits and aggregated then [45]. This leads to a more stable performance estimation. A very well known method to do so is the so called 'k-fold cross-validation'.

'K-fold cross-validation' divides the available training data into k groups of approximately equal size. By using only k-1 of these folds for the training of the model, the trained model can then be evaluated on the remaining fold - the so called held-out fold. Based "on the observations in the held-out fold" [[38], p. 181] any metric can then be calculated. This process is repeated until

each unique fold has been the held-out fold once. This results in k estimates of a metric, whereby the final k-fold cross validation estimation for this metric equals the average of the k values [38]: $CV(k) = \frac{1}{k}\sum_{i=1}^{k} \text{Metric}_i$
The process of the 'k-fold cross validation' method is illustrated in figure 15. In the top of the figure the whole data is displayed. In the first step of the cross validation, the data is shuffled and partitioned into five equally sized folds. In the first of the five iterations, the first fold is used as held-out fold, while the folds 2, 3, 4 and 5 are used to train the model. The fitted model is then evaluated with the observations from the first fold, which results in Metric_1. The same process is repeated for the remaining iterations 2-5. The only difference is the fold that is used as held-out fold for the evaluation. Each of these iterations results in a estimated metric - Metric_1, ..., Metric_5. The final estimation of the metric equals the average of the metric over all five iterations: $\frac{1}{5}\sum_{i=1}^{5} \text{Metric}_\text{i}$

würd den satz umformulieren , bin da warum auch immer hängen geblieben

in an estimated

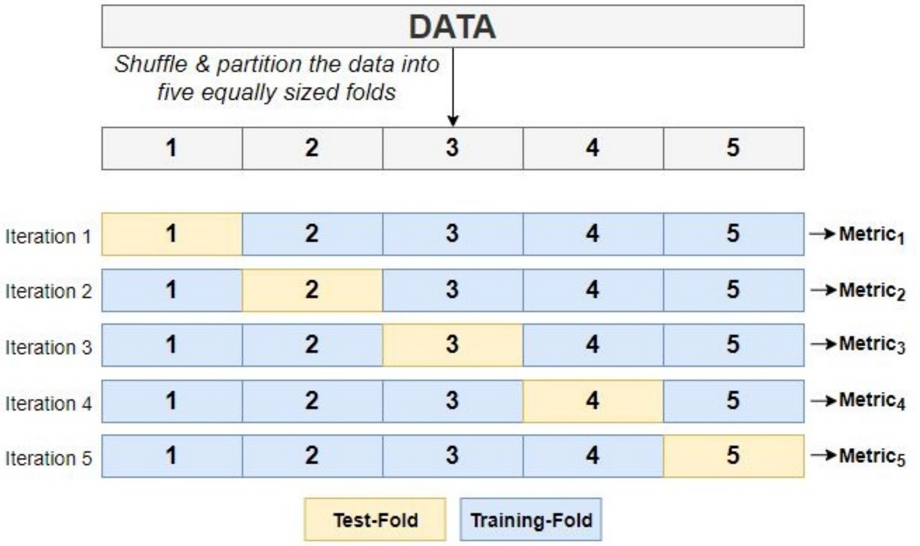


Figure 15: Example for 5-fold cross validation.

In summary: 'K-fold cross validation' is a technique to estimate the predictive performance of an approach on unseen data. It can be applied if there is only training-set, but no test-set.

3.2 Data

In this subsection the data sets from the diverse sources are investigated. In total there are **2** | **3** diverse sources of data used for the comparison of the different approaches. The comparison of approaches with an insufficient number of data sets leads to underpowered results, as the performance of the approaches across the different data sets can be highly variable [37]. Therefore enough data sets from different sources are needed for a meaningful comparison of the approaches. In the coming sections the different sources and corresponding data sets are introduced in more detail, as well as their different structures of block-wise missingness. After the introduction of each source, the technique for the evaluation is explained.

3.2.1 TCGA Data

The first source for multi-omics data is 'The Cancer Genome Atlas' (TCGA) that provides "real multi-omics data sets [...], where each of these data sets contains the measurements of patients with a certain cancer type" [[14], p. 14]. The data was not directly accessed via TCGA, but provided R. Hornung who has used the data in one of its published articles already [14]. In total 21 different data sets were provided, whereby these data sets have already been processed. The processing included the imputation of "missing values in the clinical block" [[14], p. 14] and the transformation of categorical feature variables into binary numerical features. For the imputation of the clinical covariates the 'k nearest neighbours imputation'/ 'univariate logistic regression' has been used - further details can be found in the article itself [14]. Hence the 21 provided data sets do not miss any values and only consist of numerical covariates.

of his ?

The 21 data sets actually have a survival outcome, but as this thesis aims to compare classification approaches this outcome is not used as target variable. Instead the covariate 'gender' from the 'Clinical' feature-block is used as binary categorical response variable. This is not an unusual procedure and has been applied in other studies already - e.g. [46]. Even though the 'gender' variable "is not a clinically meaningful outcome in biomedical applications, it features major advantages for a purely methodological investigation" [[46], p. 5]. Seven of the 21 available data sets do not contain a 'gender' covariate and had to be removed, such that a total of 14 usable data sets remained.

Each of these 14 remaining data sets is fully observed and consists of the same five feature-blocks. Even though the amount of available covariates for each feature-block differ between the 14 data sets, they are still in a similar field. The average amount of covariates over the 14 data sets for each feature-block is displayed in table 3.

Feature-Block	**Average amount of covariates**
Clinical	3.5
miRNA	770
Mutation	16218
CNV	57964
RNA	23559

Table 3: The average amount of covariates in each feature-block over the 14 TCGA data sets.

While the 'Clinical' feature-block represents clinical information like 'weight', 'height' and 'smoking status' the remaining four feature-blocks are different types of omics data and represent biological properties. The 'Clinical' feature-block has the lowest number of covariates on average, while the 'CNV' block has by far the most. The average amount of observations for the 14 data sets is ∼280 and hence the amount of features is on average much higher than the amount of observations - a very common property of multi-omics data.

Reducing the dimensionality of the omics feature-blocks

A problem that comes with the choice of 'gender' as target variable are the "overly strong biological signal[s]" [[46], p. 5] contained in the different omics feature-blocks. It is aimed to reduce these biological signals by only using a subset of variables for each omics block and "thus make it comparable to signals observed in applications of clinical relevance" [[46], p. 5]. Another reason for using only a subset of the available covariates per feature-block is the computational expense of evaluating models on such high dimensional data. The reduction of the computational effort is especially important for the 'fold-wise' approach from section 2.7. This approach was implemented on basis of the 'simpleRF' package [47] in plain R, as no package offered a method to prune the single trees of a random forest model. Therefore this approach is computational much slower than the other approaches from sections 2.3 - 2.6, as these were implemented with the 'randomForestSRC' package [32] that directly builds up on 'Java' and 'C'.
To find a reasonable subset for each omics feature-block, the performance of a random forest model is evaluated on each of these single blocks and their corresponding subsets. The predictive performance of the models on the different omics blocks is evaluated with 5-fold cross validation and rated with the accuracy - general details to metrics/ cross validation in the section 3.1.1/ 3.1.2. The predictive performance of a random forest model on the

single omics blocks and their corresponding subsets - based on all 14 available TCGA data sets - is displayed in figure 16.

The figure consists of totally four plots, whereby each plot represents a single omics feature-block. Each of these plots contain eight boxplots representing the accuracy - y-axis - of the model for different subsets of used covariates - x-axis. The predictive performance for the feature-blocks 'CNV' and 'RNA' is in general the best and for the feature-blocks 'miRNA' and 'Mutation' it is much worse. The feature-blocks 'miRNA' and 'Mutation' are therefore only trimmed to reduce the computational effort. As the 'miRNA' block only consists of 770 covariates on average, only 50% of the available covariates are removed. The feature-block 'Mutation' has 16218 covariates on average from which 90% are removed. While the trimming of these two feature-blocks reduces the computational effort, it does not reduce the predictive quality of these single feature-blocks too much. The amount of covariates in the 'CNV' and 'RNA' feature-blocks have to be trimmed to reduce the biological signals in these blocks. 85% of available covariates were removed from the 'RNA' block. This leads to a lower but still reasonable predictive performance with this block. The 'CNV' block leads to the best predictive performances and seems to contain a lot of strong biological signals. Therefore 95% of the available covariates from this feature-block were removed. The predictive performance with the trimmed 'CNV' block is still by far the best and not much worse than on the complete 'CNV' feature-block. The same plot but with the F-1 score as metric is displayed in figure 18 in the attachment - the results are about the same.

while the feature blocks x&y are much worse ?

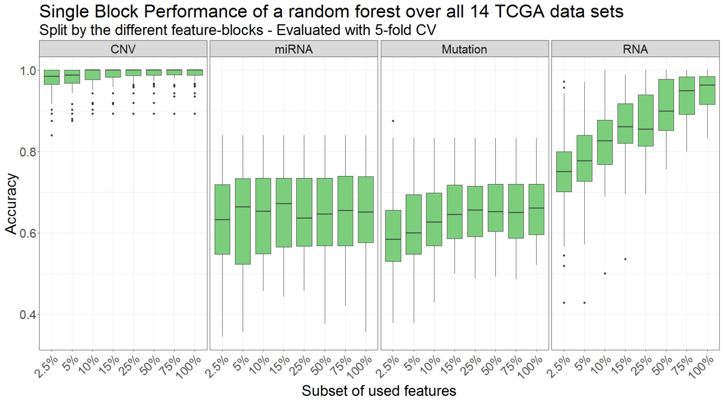


Figure 16: The accuracy of a random forest model evaluated on the single omics feature-blocks for a range of possible subsets.

Removing 50% of the covariates from the 'miRNA' block, 90% from the 'Mutation' block, 85% from the 'RNA' block and 95% from the 'CNV' block leads to data sets with much lower dimensions than originally. But nonetheless the important multi-omics property of more observed covariates than observations is still valid for the reduced data sets. The average amount of covariates over the 14 data sets for each reduced feature-block is displayed in table 4.

Reduced Feature-Block	**Average amount of covariates**
Clinical	3.5
miRNA	385
Mutation	1616
CNV	2898
RNA	3555

Table 4: The average dimensionality of the five trimmed feature-blocks for the 14 TCGA data set.

Inducing block-wise missingness

The 14 provided TCGA data sets are completely observed and do not contain any missing values. But as these data sets shall be used for the investigation of different approaches capable to deal with block-wise missingness, block-wise missingness needs to be induced into these data sets.
Each of the 14 TCGA data sets consists of the response variable 'Y' and the five feature-blocks 'Clinical', 'CNV', 'RNA', 'Mutation' and 'miRNA'. These 14 data sets are induced with block-wise missingness according to the following patterns. In the tables of these patterns a red cross indicates a missing feature-block for a fold, while a green tick indicates an observed feature-block.

Pattern 1 The first pattern of block-wise missingness is displayed in table 5. Every fold consists of the same amount of observations and to each of them there is an observed response variable 'Y', the 'Clinical' feature-block and one additional omics feature-block. Therefore each single fold consists of two feature-blocks as observed covariates - one 'Clinical' and one omics block. The 'Clinical' block is the same for all folds, while the omics blocks differ from fold to fold.

	Y	Clinical	CNV	RNA	Mutation	miRNA
Fold 1	✓	✓	✓	✗	✗	✗
Fold 2	✓	✓	✗	✓	✗	✗
Fold 3	✓	✓	✗	✗	✓	✗
Fold 4	✓	✓	✗	✗	✗	✓

Table 5: The first block-wise missingness pattern for the TCGA data sets.

Pattern 2 The second pattern of block-wise missingness is displayed in table 6. The amount of observations in the different folds is equal for all folds, while the amount of observed feature-blocks differ from fold to fold. The order of the feature-blocks was randomly changed in comparison to 'Pattern 1', whereby the last omics block 'CNV' is observed for all folds and the first omics block 'miRNA' only for the first fold. The amount of available feature-blocks decreases from 'Fold 1' to 'Fold 4'. 'Fold 1' is completely observed within all five feature-blocks and 'Fold 4' in only two feature-blocks - 'Clinical' and 'CNV'.

	Y	Clinical	miRNA	RNA	Mutation	CNV
Fold 1	✓	✓	✓	✓	✓	✓
Fold 2	✓	✓	✗	✓	✓	✓
Fold 3	✓	✓	✗	✗	✓	✓
Fold 4	✓	✓	✗	✗	✗	✓

Table 6: The second block-wise missingness pattern for the TCGA data sets.

Pattern 3 The third pattern of block-wise missingness is displayed in table 7. The amount of observations is equally split to the four folds. For each of these folds it is randomly drawn which feature-blocks are observed. The probability for any feature-block to be observed equals $\frac{2}{3}$ and the probability of not observing a block therefore $\frac{1}{3}$. The 'Clinical', as well as the 'RNA' feature-block were sampled for every fold. Therefore the four folds only differ in the feature-blocks 'CNV', 'Mutation' and 'miRNA'. The same feature-blocks were sampled for 'Fold 3' and 'Fold 4', such that these folds have the exact same covariates. Therefore these two folds are actually a single fold and the data consists of only three unique folds.

	Y	Clinical	CNV	RNA	Mutation	miRNA	
Fold 1	✓	✓	✓	✓	✗	✓	
Fold 2	✓	✓	✓	✓	✗	✗	
Fold 3	✓	✓	✗	✓	✓	✓	}**Fold3**
Fold 4	✓	✓	✗	✓	✓	✓	

Table 7: The third block-wise missingness pattern for the TCGA data sets.

Pattern 4 The fourth and last pattern of block-wise missingness is displayed in table 8. Other than in the three patterns before, the amount of observations is equally split in two instead of four folds. The next difference to the first three patterns is that the amount of feature-blocks is reduced from five to three. For this two of each of the four available omics feature-blocks were combined into a single omics feature-block - this was done completely at random. The feature-blocks 'RNA' and 'miRNA' were combined to a single feature-block, as well as the feature-blocks 'Mutation' and 'CNV'. Therefore the whole data consists of three feature-blocks - 'Clinical', 'RNA & miRNA' and 'Mutation & CNV' - and two folds.

this has been
done randomly

	Y	Clinical	RNA & miRNA	Mutation & CNV
Fold 1	✓	✓	✓	✗
Fold 2	✓	✓	✗	✓

Table 8: The fourth block-wise missingness pattern for the TCGA data sets.

Evaluation Technique

The 14 data sets from the TCGA source are used to asses the predictive performance of the different approaches on data with block-wise missingness. All data sets are completely observed and to none of these exists a separate test set. A slightly modified version of 5-fold cross validation is applied to asses the predictive performance of the different approaches on this data. The process is illustrated in algorithm 3.
In the very beginning a data-set D, an approach App and a block-wise missingness pattern $Patt$ have to be selected. The data-set D is split into five equally sized folds, whereby a single fold is used as test-set and the remaining four folds as training-set. Then block-wise missingness is induced into the train-set according to the selected pattern of missingness $Patt$ and the model is fitted on this data. The predictive performance of an approach

vllt manchmal synonym verwenden oder satz umformulieren, sodass du auf test-set verzichten kannst

does not only depend on the training-set, but also on the available feature-blocks in the test-set. Hence the approach *App* is evaluated not only on a fully observed test-set, but also on a test-set with different combinations of missing feature-blocks. This is relevant, as block-wise missingness can affect the test-set as well as the training data. Therefore the predictive performance of the fitted model is evaluated on all possible combinations of observed feature-blocks in the test-set. Firstly the approach is evaluated on a test-set with all feature-blocks available (2.3.1). Then it is further evaluated on a all possible test-sets with one missing feature-block (2.3.2) - e.g. test-set with missing 'CNV' feature-block. Following it is evaluated on all possible test-sets with two missing feature-blocks (2.3.3) - e.g. test-set with missing 'CNV' & 'RNA' feature-block. Subsequently it is evaluated on all possible test-sets with three missing feature-blocks (2.3.4) - e.g. test-set with missing 'CNV' & 'RNA' & 'miRNA' feature-block. And finally it is evaluated on the test-sets with only a single observed feature-block (2.3.5) - e.g. test-set has only the 'CNV' feature-block. For each of these possible test-sets the metrics are calculated. When the evaluation of an approach *App* on a data-set *D* with the block-wise missingness pattern *Patt* is done, to each possible test-set there are five values of each metric. The average of these metrics for each possible test-set are then used to compare the predictive performance of the different approaches in section 4.1.

Algorithm 3: Evaluation of the approaches with the TCGA data

```
Input  : $D \leftarrow$ TCGA data-set with n observations & p features
         $App \leftarrow$ Applied approach
         $Patt \leftarrow$ Pattern of block-wise missingness
1. Split the fully observed data-set D into five equally sized folds
2. for $k \leftarrow 1$ to $5$ do
    2.1 Use fold $k$ as held-out fold & the remaining four folds as
      training-set;
    2.2 Induce the block-wise missingness according to $Patt$ into the
      training-set;
    2.3 Evaluate the predictive performance of the approach $App$ for
      different combinations of observed feature-blocks in the test-set;
        2.3.1 Evaluation on the fully observed test-set;
        2.3.2 Evaluation on test-sets with one missing feature-block;
        2.3.3 Evaluation on test-sets with two missing feature-blocks;
        2.3.4 Evaluation on test-sets with three missing feature-blocks;
        2.3.5 Evaluation on test-sets with a single observed
      feature-block;
```

In summary: To evaluate an approach on a TCGA data-set 5-fold cross validation is applied. The training data is induced with a block-wise missingness pattern and the model is fitted on this data. The model is then not only evaluated on the fully observed test-set, but also for all possible combinations of block-wise missingness in the test-set.

3.2.2 Data from Hagenberg

3.2.3 Real data

4 Results

4.1 Own data

4.1.1 Scenario 1

4.1.2 Scenario 2

4.1.3 Scenario 3

4.1.4 Scenario 4

4.2 Data from Hagenberg

4.3 Real data

5 Discussion and Conclusion

6 Bibliography

[1] Francis S Collins. "Medical and societal consequences of the Human Genome Project". In: *New England Journal of Medicine* 341.1 (1999), pp. 28–37.

[2] *National Human Genome Research Institute.* `https://www.genome.gov/about-genomics/fact-sheets/Sequencing-Human-Genome-cost`. Accessed: 2020-01-07.

[3] Belinda JF Rossiter and C Thomas Caskey. "Impact of the Human Genome Project on medical practice". In: *Annals of surgical oncology* 2.1 (1995), pp. 14–25.

[4] Sara Goodwin, John D McPherson, and W Richard McCombie. "Coming of age: ten years of next-generation sequencing technologies". In: *Nature Reviews Genetics* 17.6 (2016), p. 333.

[5] *Veritas - The Genome Company.* `https://www.veritasgenetics.com/myGenome`. Accessed: 2020-01-19.

[6] Ke Bi et al. "Transcriptome-based exon capture enables highly cost-effective comparative genomic data collection at moderate evolutionary scales". In: *BMC genomics* 13.1 (2012), p. 403.

[7] Anne-Laure Boulesteix et al. "IPF-LASSO: Integrative-penalized regression with penalty factors for prediction based on multi-omics data". In: *Computational and mathematical methods in medicine* 2017 (2017).

[8] Valeria D'Argenio. "The high-throughput analyses era: are we ready for the data struggle?" In: *High-throughput* 7.1 (2018), p. 8.

[9] Gregory B Gloor et al. "Microbiome profiling by illumina sequencing of combinatorial sequence-tagged PCR products". In: *PloS one* 5.10 (2010).

[10] Shrutii Sarda and Sridhar Hannenhalli. "Next-generation sequencing and epigenomics research: a hammer in search of nails". In: *Genomics & informatics* 12.1 (2014), p. 2.

[11] Forest M White. "The potential cost of high-throughput proteomics". In: *Sci. Signal.* 4.160 (2011), pp. 8.

[12] *National Institutes of Health.* `https://commonfund.nih.gov/arra/highthroughput`. Accessed: 2020-01-30.

[13] Moritz Herrmann. "Large-scale benchmark study of prediction methods using multi-omics data". PhD thesis. 2019.

[14] Roman Hornung and Marvin N Wright. “Block Forests: random forests for blocks of clinical and omics covariate data”. In: *BMC bioinformatics* 20.1 (2019), p. 358.

[15] Simon Klau et al. “Priority-Lasso: a simple hierarchical approach to the prediction of clinical outcome using multi-omics data”. In: *BMC bioinformatics* 19.1 (2018), p. 322.

[16] Stefanie Hieke et al. “Integrating multiple molecular sources into a clinical risk prediction signature by extracting complementary information”. In: *BMC bioinformatics* 17.1 (2016), p. 327.

[17] Qing Zhao et al. “Combining multidimensional genomic measurements for predicting cancer prognosis: observations from TCGA”. In: *Briefings in bioinformatics* 16.2 (2015), pp. 291–303.

[18] Roman Hornung et al. “Random forests for multiple training data sets with varying covariate sets”. manuscript - unpublished yet. - in prep.

[19] Norbert Krautenbacher. “Learning on complex, biased, and big data: disease risk prediction in epidemiological studies and genomic medicine on the example of childhood asthma”. PhD thesis. Technische Universität München, 2018.

[20] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media, 2009.

[21] Jonas Hagenberg. “Penalized regression approaches for prognostic modelling using multi-omics data with block-wise missing values”. manuscript - unpublished yet. - in prep.

[22] Hemant Ishwaran et al. “Random survival forests”. In: *The annals of applied statistics* 2.3 (2008), pp. 841–860.

[23] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[24] *What is the difference between Bagging and Boosting?* `https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/`. Accessed: 2020-03-06.

[25] Leo Breiman et al. *Classification and regression trees.* CRC press, 1984.

[26] Wenbin Lu. *Lecture 21: Classification and Regression Trees.* Department of Statistics North Carolina State University, 2019.

[27] Bernd Bischl and Christoph Molnar. *Introduction to Machine Learning - Chapter 13: Trees.* Department of Statistics - LMU Munich, WinterTerm 2017/18.

[28] Terry Therneau and Beth Atkinson. *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-15. 2019. URL: https://CRAN.R-project.org/package=rpart.

[29] Leo Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[30] *Bootstrap Sample: Definition, Example*. https://www.statisticshowto.com/bootstrap-sample/. Accessed: 2020-03-04.

[31] Bernd Bischl and Christoph Molnar. *Introduction to Machine Learning - Chapter 15: Bagging and Random Forests*. Department of Statistics - LMU Munich, WinterTerm 2017/18.

[32] H. Ishwaran and U.B. Kogalur. *Fast Unified Random Forests for Survival, Regression, and Classification (RF-SRC)*. R package version 2.9.3. manual, 2020. URL: https://cran.r-project.org/package=randomForestSRC.

[33] Daniel J Stekhoven and Peter Bühlmann. "MissForest—non-parametric missing value imputation for mixed-type data". In: *Bioinformatics* 28.1 (2012), pp. 112–118.

[34] Olga Troyanskaya et al. "Missing value estimation methods for DNA microarrays". In: *Bioinformatics* 17.6 (2001), pp. 520–525.

[35] Gary King et al. "Analyzing incomplete political science data: An alternative algorithm for multiple imputation". In: *American political science review* 95.1 (2001), pp. 49–69.

[36] James Honaker, Gary King, and Matthew Blackwell. "Amelia II: A Program for Missing Data". In: *Journal of Statistical Software* 45.7 (2011), pp. 1–47. URL: http://www.jstatsoft.org/v45/i07/.

[37] Anne-Laure Boulesteix, Sabine Lauer, and Manuel JA Eugster. "A plea for neutral comparison studies in computational sciences". In: *PloS one* 8.4 (2013).

[38] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[39] *Performance Metrics for Classification problems in Machine Learning*. https://medium.com/thalus-ai/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b. Accessed: 2020-04-22.

[40] Mohammad Hossin and MN Sulaiman. "A review on evaluation metrics for data classification evaluations". In: *International Journal of Data Mining & Knowledge Management Process* 5.2 (2015), p. 1.

[41] Sabri Boughorbel, Fethi Jarray, and Mohammed El-Anbari. "Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric". In: *PloS one* 12.6 (2017).

[42] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[43] *Balanced accuracy: what and why?* `http://mvpa.blogspot.com/2015/12/balanced-accuracy-what-and-why.html`. Accessed: 2020-04-23.

[44] Pierre Baldi et al. "Assessing the accuracy of prediction algorithms for classification: an overview ". In: *Bioinformatics* 16.5 (May 2000), pp. 412–424. ISSN: 1367-4803. DOI: `10.1093/bioinformatics/16.5.412`. eprint: `https://academic.oup.com/bioinformatics/article-pdf/16/5/412/476945/160412.pdf`. URL: `https://doi.org/10.1093/bioinformatics/16.5.412`.

[45] Bernd Bischl and Christoph Molnar. *Introduction to Machine Learning - Chapter 9: Performance Estimation.* Department of Statistics - LMU Munich, WinterTerm 2017/18.

[46] Nicole Schüller et al. "Improved outcome prediction across data sources through robust parameter tuning". In: (2019).

[47] Marvin N Wright. *SimpleRF*. `https://github.com/mnwright/simpleRF`. 2018.

7 Attachment

Figures

Impurity functions (3), (4) and (5) for a binary target variable

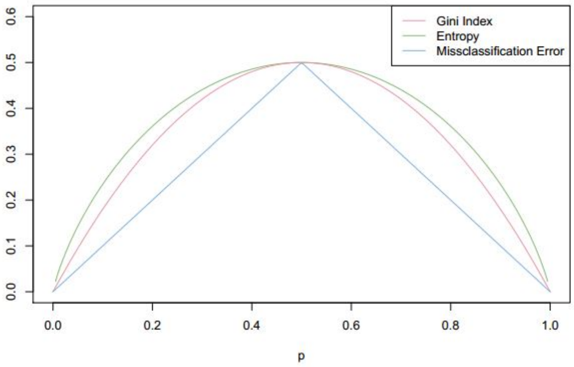


Figure 17: The different impurity functions (3), (4) and (5) plotted for a given fraction of a binary target variable within any node N [[27], p. 13]

.

Predictive performance of a random forest model on the single feature-blocks of the 14 TCGA data sets

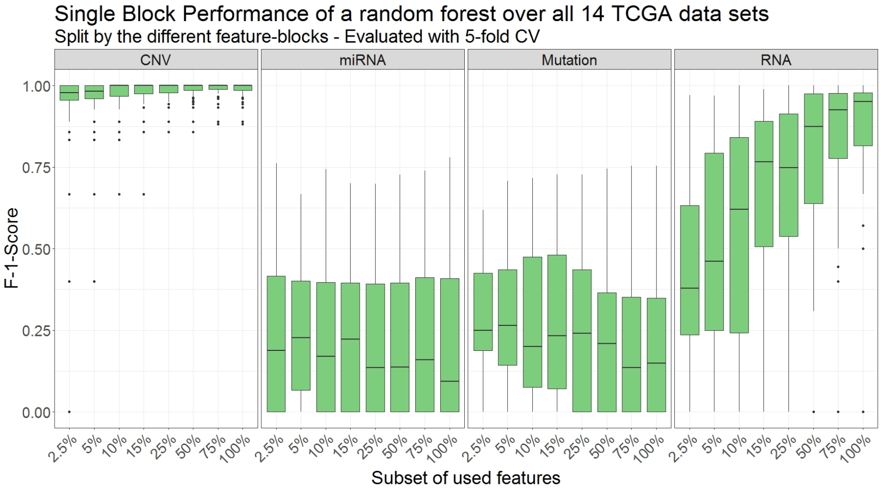


Figure 18: The F1-Score of a random forest model evaluated on the single omics feature-blocks for a range of possible subsets. The results were obtained on basis of the 14 TCGA data sets.

Tables