RADBOUD UNIVERSITY

MASTER'S THESIS IN ARTIFICIAL INTELLIGENCE

Combining CT scans and clinical features for improved automated COVID-19 detection

---

Author:
Roel HACKING, s1047200

Supervisor 1:
Dr. Max HINNE
Supervisor 2:
M.Sc Luuk BOULOGNE

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science

in the

Diagnostic Image Analysis Group
AI Department

September 1, 2021

RADBOUD UNIVERSITY

Abstract

Faculty of Social Sciences
AI Department

Master of Science

Combining CT scans and clinical features for improved automated COVID-19 detection

by Roel HACKING, s1047200

During the first peak of the COVID-19 pandemic, hospitals in hard-hit regions were overflowing with patients at the emergency unit with respiratory complaints. Since the RT-PCR test was in limited supply at the time and test results took a long time to obtain, many hospitals opted to use chest CT scans of COVID-19 suspects. As a result of this, several studies examined the possibility of automating the detection of COVID-19 in CT scans. One such study, by Lessmann et al., 2020, developed a model to predict COVID-19 severity scores based on these chest CT scans. In this thesis, we extended their model in several ways to take into account additional clinical values (such as blood values, sex, and age) to predict either PCR outcomes or clinical diagnoses. Based on data from the Canisius-Wilhelmina Ziekenhuis (CWZ) hospital and Radboudumc hospitals, as well as the COVID-19 dataset by Ning et al., 2020, we found that integrating these two modalities can indeed lead to improved performance when both clinical and visual features are of sufficient quality. When training on data from the CWZ hospital and evaluating on data from the Radboudumc hospital, models using only clinical features or visual features achieved Area Under the ROC Curve (AUC) values of 0.773 and 0.826, respectively; their combination resulted in an AUC of 0.851. Similarly, when training on data from the Union hospital in the iCTCF dataset and predicting on data from the Union hospital in that same dataset, we obtained AUCs of 0.687 and 0.812 for clinical and visual features, respectively; their combination resulted in an AUC of 0.862.

However, we also discovered that the patterns of missing data present in these clinical feature datasets can play an essential role in the performance of the models fitted on them. We thus developed additional methods to analyze and mitigate this effect to obtain fairer evaluations and increase model generalizability. Still, the high diagnostic performance of some of our models suggests that they could be adapted into clinical practice, and our methods pertaining to missing data could be used to aid further research using clinical feature datasets.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and background

For more than a year now, the world has been under siege by the novel coronavirus, SARS-CoV-2. This virus, which induces a respiratory disease named Coronavirus Disease 2019 (Covid-19) in humans (as well as other animals), has been putting healthcare systems around the world under tremendous pressure (Hui et al., 2020; Verelst, Kuylen, and Beutels, 2020; Wu, Leung, and Leung, 2020; Wu et al., 2020). As COVID-19 cases surge, local intensive care units struggle to care adaquately for those who suffer most severely from the disease (Barrasa et al., 2020; Uppal et al., 2020). In addition, the most commonly used COVID-19 test—a reverse-transcription polymerase chain reaction (RT-PCR) test—was often in short supply and initially took many hours to process, obstructing the immediate treatment and correct placement of patients in corona wards (Xie et al., 2020). As such, hospitals would employ alternative diagnostic methods. Specifically, they would often use Computed Tomography (CT) scans of the suspect's chest (Ye et al., 2020).

These chest CT scans can be analyzed by radiologists, who can subsequently generate quick working diagnoses with decent accuracy; the study by Bai et al., 2020 suggested an accuracy of 90% on the task. Various institutions have attempted to automate this process by utilizing state-of-the-art machine learning techniques to classify these chest CT scans (Vaishya et al., 2020; Li et al., 2020; Bai et al., 2020; Jin et al., 2020; Lessmann et al., 2020). These techniques have generally achieved similar performance to radiologists.

With the emergence of new tests, colloquially referred to as "rapid tests", that can classify COVID-19 much faster than a standard PCR test, the necessity for chest CT-based diagnosis in intensive care units has decreased (Crozier et al., 2021). Nevertheless, automated chest CT-based COVID-19 detection remains relevant in three ways. Firstly, since inference—making a prediction—for most machine learning models is relatively fast and involves no manual operation by humans, these models could still be employed as background tests whenever a chest CT scan is performed in a hospital. If the model indicates it is likely that the patient suffers from COVID-19, the operator(s) can be alerted and a more conclusive COVID-19 clinical diagnosis can be made.

Secondly, while most of these systems currently only perform diagnosis, they could be expanded to perform more complex tasks, such as providing prognosis, progression, or suggested treatment of patients in such a way as to minimize the mortality rate.

Finally, the machine learning techniques developed can be applied more broadly. While this thesis mostly considers their usage for COVID-19 classification, they could also be adapted for different respiratory diseases. Even more generally, they might generalize to different medical imaging domains entirely or even nonmedical domains.

In addition to CT scans, other clinical parameters—such as sex, age, and blood values—might also be valuable. These clinical features can provide additional context when evaluating CT scans, which might allow for more robust COVID-19 classification.

Therefore, in this thesis, we considered various approaches to enhance COVID-19 diagnosis from chest CT, taking into account related clinical parameters. In addition, we consider the kinds of problems this presents and what general methods could be utilized to remedy such issues. Part of the work in this thesis builds upon the work by Lessmann et al., 2020, which is further elaborated upon in section 1.7. We make three main novel contributions. Firstly, this thesis is the first to consider the combination of neural networks and tree-based learning techniques on COVID-19 CT and clinical feature datasets. Secondly, this thesis is the first to combine CT scans and clinical features for a dataset obtained at Radboudumc. Finally, this is the first work to consider the effects of missing data in clinical feature data, specifically in terms of model generalizability. For this thesis, we posed the research question:

> *"Can chest CT-scans and clinical features be combined to generate COVID-19 diagnoses with greater accuracy and generalizability than either modality on its own?"*

with the following subquestions:

1. How accurately can COVID-19 be predicted based on clinical features alone?
2. How can clinical and visual features be combined to improve COVID-19 prediction?
3. Can models trained on COVID-19 data from one hospital generalize to data from a different hospital?
4. Are the patterns of missing data in COVID-19 clinical data predictive of the diagnosis?
5. Do the patterns of missing data in COVID-19 clinical data harm model generalizability?
6. How can overfitting on patterns of missing data be mitigated?

The following sections will provide further background on SARS-CoV-2, the disease it brings about, the relevant machine learning techniques—in particular, neural networks and tree-based techniques—for this thesis, as well as the datasets that were utilized. The final section discusses the machine learning model of Lessmann et al., 2020.

1.1 Machine Learning Techniques

Machine Learning (ML) is the study of computer algorithms that improve themselves automatically. In supervised ML, the computer is given some 'training' dataset $X \in \mathbb{R}^{n \times m}$, where n is the number of samples or instances and m is the number of input values or features of each sample. The computer is then tasked to learn to predict some vector $y \in \mathbb{R}$, where each instance i has some associated target label y_i. The model that is then obtained can hopefully generalize to some new, unseen X, where the target vector y is not known.

The target label vector y can take two forms. Firstly, they can be classes. For example, if we are training an ML model to recognize hand-written digits, there would be ten different possible classes for each y_i. The process for predicting y is then referred to as classification. The target labels can also take the form of continuous values. For example, if we are training an ML model to predict housing prices based on some features of each house, each y_i would be a (mostly continuous) price. The process of predicting y is in this case referred to as regression.

There are several different metrics one can use to evaluate the performance of such ML models. This thesis is most concerned with the classification metric known as the area under the curve (AUC). This metric is based on the receiver operating characteristic curve (ROC), which is generated by plotting the true positive rate against the false positive rate at various thresholds of the classifier's output. The AUC is then computed as the area under this ROC curve. The metric value lies between 0 and 1, and a higher value indicates better performance.

There are many techniques that attempt to accomplish this. The following few sections will introduce the various machine learning techniques that are relevant to this thesis. Some of these techniques are relevant for CT image classification, whereas tree-based techniques are more useful when operating on structured data, such as clinical features. Finally, the last subsection discusses missing data, which is a common problem in many types of data analysis and is also present in the data considered in this thesis.

1.1.1 Artificial Neural Networks

While first considered back in 1957 (Rosenblatt, 1957), artificial neural networks (ANN) have only relatively recently been making breakthroughs in many problem domains that previously remained nearly impossible for computers to solve reliably. This is in part the result of increased highly-parallelized computational power in the form of Graphical Processing Units (GPU) and significantly larger amounts of data available to researchers. The first significant breakthrough was the introduction of convolutional neural networks (CNN) in 2011, resulting in the first-ever human-competitive image classification systems (Yan, Yoshua, and Geoffrey, 2015).

ANNs are computational models that consist of basic components called artificial neurons. These neurons receive some input $x \in \mathbb{R}^m$, and together with a bias $b \in \mathbb{R}$ and weights $w \in \mathbb{R}^m$, generate an output $y \in \mathbb{R}$ (Emmert-Streib et al., 2020). This is generally computed as follows:

$$y = \phi(z) = \phi(w^T x + b), \tag{1.1}$$

where ϕ is some nonlinear activation function. This function is required to make the ANN nonlinear, which in turn allows it to have a nonlinear decision boundary. This nonlinear decision boundary is necessary to model targets that vary non-linearly with the inputs.

These basic units are then grouped together in layers where, in a basic feed-forward neural network, each neuron receives input from all neurons in the previous layer. A simple example of such a fully connected feed-forward network can be seen in figure 1.1. The first layer is referred to as the input layer and represents all m features in X. The last layer is referred to as the output layer. All layers in between are referred to as hidden layers.

By iterating over each layer and computing the values of each neuron as in equation 1.1, one can obtain the values of the neurons in the output layer of the network.

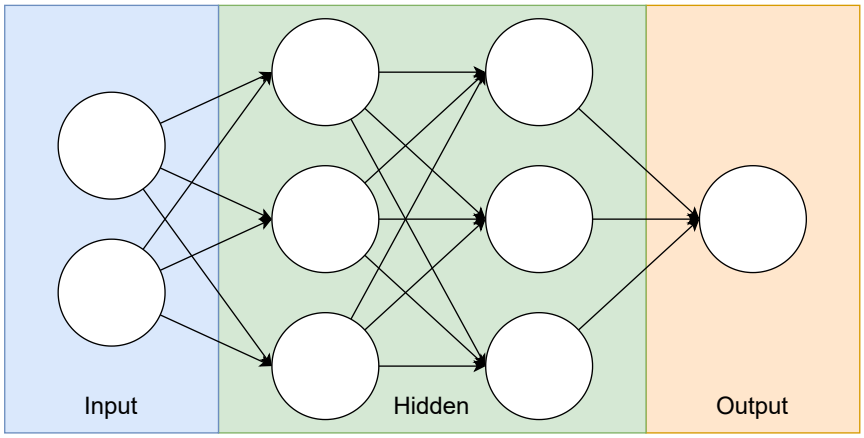


FIGURE 1.1: A diagram of a simple fully-connected feed-forward artificial neural network with two input neurons, one output neuron, and two hidden layers both containing two hidden neurons.

The output can then be evaluated using a loss function. One such loss function commonly used for classification is binary cross-entropy:

$$H_p(q) = \frac{1}{N}\sum_{i=1}^{N} y_i * \log(p(y_i)) + (1 - y_i) * \log(1 - p(y_i)),$$

where N is the number of samples, $p(y)$ is the predicted probability of y, and y_i is the ground-truth target label of sample i.

There are many methods that allow one to find the weights for these networks, such as evolutionary algorithms or backpropagation. However, the latter is the most commonly used method for supervised learning (Rumelhart, Hinton, and Williams, 1986). Backpropagation works by computing the gradient of the loss with respect to each weight in the network by means of the chain rule. These gradients can then be used to perform gradient descent to minimize the loss function by repeatedly adjusting the weights, improving the network's performance.

Convolutional Neural Networks

While it is possible to use the previously described ANNs to perform computer vision tasks such as image classification by simply giving each input pixel as an input to the network, there are much more effective approaches. Specifically, most modern image classification networks are convolutional neural networks (CNN).

These CNNs, in addition to incorporating fully connected layers, also contain convolutional layers. In such layers, instead of each neuron being connected to all neurons in the next layer, neurons are only connected to those close to them. This significantly reduces the number of parameters of the network. In addition, all connections between local receptive fields and neurons use the same set of weights, and this set of weights is denoted as a kernel. Expressed differently, each convolutional layer consists of a set of convolution kernels, where each weight in the kernel is an optimizable parameter of the network. The activations of the layer are simply the convolutions of the previous layer with the kernels of the current layer. Convolution is shown graphically in figure 1.2.

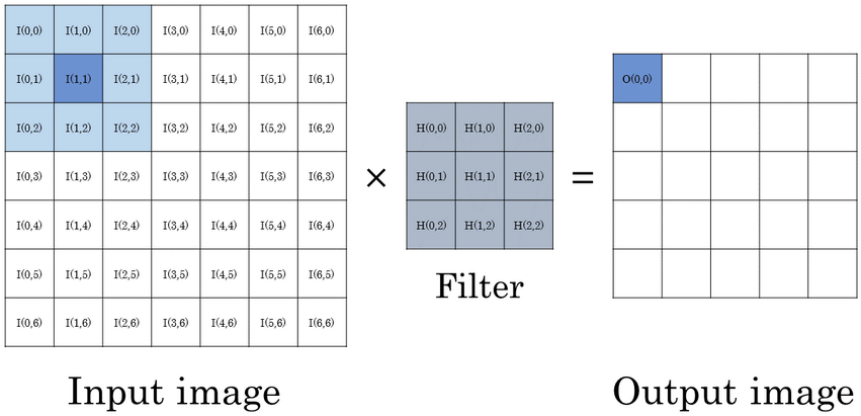


FIGURE 1.2: Example of convolution (adapted from Baskin et al., 2018).

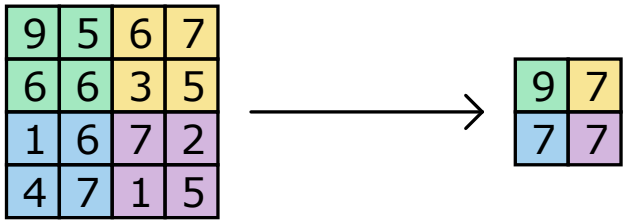


FIGURE 1.3: Example of max-pooling with a 2x2 kernel.

CNNs also incorporate pooling layers, which help reduce the input size with some pooling method. One such pooling method is max-pooling, which takes repeated maxima of the window to generate a smaller output. This operation is shown graphically in figure 1.3. Convolutional layers, pooling layers, and fully connected layers are then combined to construct a CNN.

One can then adapt the backpropagation algorithm to effectively compute the gradients for such convolutional neural networks, allowing one to optimize the network parameters in the same way as regular fully connected neural networks with (stochastic) gradient descent.

1.1.2 Tree-based learning

While (convolutional) neural networks have brought about great boosts in performance in computer vision and other domains, tree-based machine learning methods still perform very well on tabular data (such as the data shown in table 1.1). They are also often significantly easier and faster to train. As such, they are well-suited when working with data such as clinical lab values (Guelman, 2012).

Decision trees

The most basic tree-based machine learning technique is that of Decision Trees (DT) (Rokach and Maimon, 2005). A decision tree is simply a series of nodes, each of which asks some yes-no question. One starts at the tree's root node and, depending on the answer at each node, a branch is selected, and the process is repeated until a leaf node is reached, which is the model's output. An example of a decision tree fitted on the Titanic dataset can be seen in figure 1.4.

TABLE 1.1: A sample of 10 random instances in the Titanic dataset, which contains various data about passengers on the Titanic before it sank, including whether they survived or not. This dataset is often used in examples of supervised machine learning on tabular data, where one would usually attempt to predict the last column ("Survived") based on features such as sex and age.

	Pclass	Sex	Age	Fare	Cabin	Embarked	Survived
504	3	female	37.0	9.5875	?	S	0
357	1	female	22.0	55.0000	E33	S	1
490	3	male	9.0	15.9000	?	S	1
535	3	female	30.0	8.6625	?	S	0
447	2	female	13.0	19.5000	?	S	1
52	3	male	21.0	7.8000	?	S	0
574	3	female	?	7.7500	?	Q	1
384	1	female	35.0	52.0000	?	S	1
658	3	female	32.0	15.5000	?	Q	0
298	1	female	2.0	151.5500	C22 C26	S	0

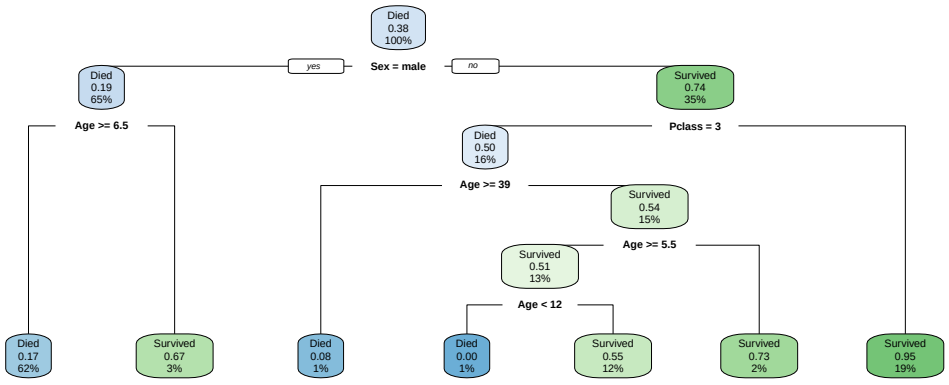


FIGURE 1.4: A simple decision tree fitted on the Titanic dataset. The depth of the tree was limited to 3 to simplify the visualization.

There are many ways to generate such decision trees. One popular algorithm is that of Classification And Regression Trees (CART, Breiman et al., 2017). This algorithm considers all features in the dataset in sequence. It then computes which features and thresholds or categories of those features help best to distinguish the samples based on their target values. For categorical features, one could, for example, split the data based on the class of a passenger, such that all first-class passengers would be assigned to one leaf and all others to the other leaf. For numerical features, one could consider all mean values between subsequent feature values as potential thresholds based on which the data could be split.

In order to evaluate the different options, some criterion is computed for each possible splitting. For classification, a common criterion is Gini impurity. This metric represents the extent to which generated leaf nodes fail to be 'pure' (i.e., only contain samples with the same classification label). The Gini impurity of a leaf node is defined as follows:

$$G = \sum_{i=1}^{C} p_i * (1 - p_i),$$

where G is the Gini impurity, p_i is the fraction of samples with target label i, and C is the total number of classes. To compute the Gini impurity across all leaf nodes resulting from a particular splitting, one can use the following equation:

$$G_T = \sum_{i=1}^{L} \frac{S_i}{S} * G_i,$$

where G_T is the weighted average Gini impurity, L is the number of leaf nodes, S_i is the number of samples in leaf node i, S is the total number of samples, and G_i is the Gini impurity of leaf node i.

Once the weighted average Gini impurities have been computed for all possible splittings, the smallest can be selected (as this best separates the classes). One can then recursively repeat this process for each leaf node, considering only the remaining features. When there are no more features available to split on, or when the Gini impurity of a node is better than the Gini impurity achieved with any remaining possible splitting, a leaf node is generated. This then results in a decision tree that can be used for inference. For a more formal description, see algorithm 1.

Decision trees are very simple to build, evaluate, and interpret. However, to quote 'The Elements of Statistical Learning' by Hastie, Tibshirani, and Friedman, 2009: "Trees have one aspect that prevents them from being the ideal tool for predictive learning, namely inaccuracy." That is, decision trees work well on the data they were trained on, but generally fail to generalize.

Random forests

There are, however, ways to utilize decision trees that allow for much greater generalization. One such method is that of Random Forests (RF) (Breiman, 1999). Random forests work by generating many decision trees.

For each decision tree, the input data used to generate it is an augmented version of the original dataset. More specifically, the data are bootstrapped. That is, a dataset the size of the original is generated by randomly sampling it (with replacement). This means that it is likely some samples will be missing in this bootstrapped dataset and, as a result, some samples will also be duplicated (on average, one would expect

Algorithm 1 Decision tree learning

Require: $D \in \mathbb{R}^{n \times m}$, where $n \in \mathbb{N}$ is the number of samples, $m \in \mathbb{N}$ is the number of features, and D_i^j refers the the jth feature of the ith sample

function DECISIONTREE(D)
  if D is pure or some other stopping criterion is met **then**
    $C \leftarrow$ most common class in D
    return $Leaf(C)$
  $split \leftarrow$ split with smallest G_T
  $G_{split} \leftarrow G_T$ of split
  $G_D \leftarrow G$ of D
  if $G_{split} < G_D$ **then**
    return $Tree(DecisionTree(split_{left}), DecisionTree(split_{right}))$
  else
    $C \leftarrow$ most common class in D
    return $Leaf(C)$

$1 - \frac{1}{e}$ of the original data to be present in the bootstrapped data, Efron and Tibshirani, 1997).

Using this bootstrapped dataset, a decision tree is generated. However, at each step, instead of considering all available features for splitting, only a random subset is considered. The process of generating trees is repeated a number of times to obtain the random forest.

In order to make a decision given a sample, we simply run each individual tree in the forest and then use the most common classification as the output of the forest. The process of bootstrapping data and aggregating data like this is referred to as bagging (Bootstrap Aggregation (Breiman, 1996)) and can also be applied to using other base learners, though decision trees are most commonly used.

As a result of both the bootstrapping and the random feature selection, the forest consists of a great variety of trees. This, in turn, is what makes random forests more effective than a lone tree. For a more formal description, see algorithm 2.

Algorithm 2 Random forest

Require: $D \in \mathbb{R}^{n \times m}$, where $n \in \mathbb{N}$ is the number of samples, $m \in \mathbb{N}$ is the number of features, and D_i^j refers the the jth feature of the ith sample, $k \in \mathbb{N}$

function RANDOMFOREST(D, k)
  $trees \leftarrow \varnothing$
  for $i \leftarrow 1$ to k **do**
    $D_B \leftarrow Bootstrap(D)$
    $T \leftarrow DecisionTree(D)$
    $trees \leftarrow trees \cup \{T\}$
  return $trees$

AdaBoost

Another approach to combining decision trees is AdaBoost (Freund, Schapire, and Abe, 1999). AdaBoost is similar to random forests, but differs in three key aspects:

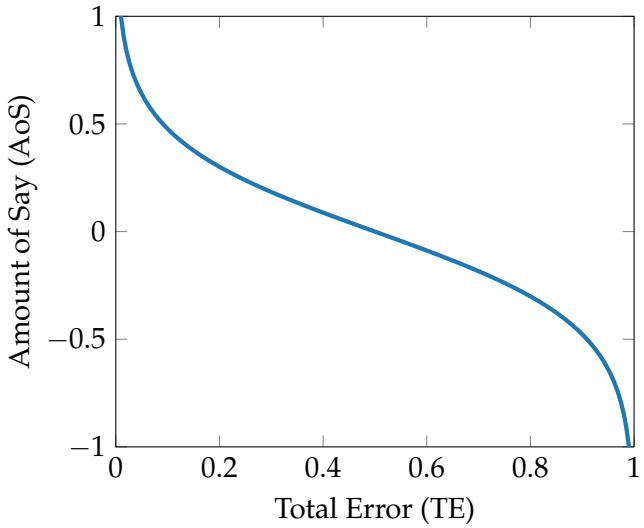


FIGURE 1.5: The Amount of Say (AoS) used in AdaBoost as a function of the Total Error (TE).

1. In random forests, each tree often has an unlimited depth. In AdaBoost, however, each tree is just a root node with two leaves (a stump), making them very weak learners.

2. In random forests, each tree is weighted equally when generating a decision. However, in AdaBoost, the amount of influence each tree has on the final outcome of the model differs.

3. In random forests, each decision tree is made independently of the others. In AdaBoost, however, the stumps are generated in sequence, and how a stump is formed depends on the errors made by the previous stump.

The algorithm works by first assigning weights to all samples in the dataset. Initially, each sample is weighted equally (i.e. $\frac{1}{n}$, where n is the number of samples). Then, a stump is generated in the same way as described in section 1.1.2. The sample weights are ignored in this first step, as they are all equal to each other at this point. Next, the weight—which is also referred to as the Amount of Say (AoS)—of this stump is determined. The AoS is determined by the total error (TE), which is the sum of the sample weights associated with samples that were classified incorrectly. The AoS is computed as follows:

$$AoS = \frac{1}{2}\log(\frac{1-TE}{TE}).$$

As the AoS approaches positive infinity, the TE goes to zero. When the TE is 0.5 (no better than random), the AoS is zero, meaning the stump will have no impact on the model's output. When the TE goes to one, the AoS will approach negative infinity, meaning the model will lean towards the opposite of what the stump suggests. See figure 1.5.

In order to correct for the mistakes made by the first stump, the sample weights are adjusted to make those that were incorrectly classified more important. Specifically, all weights of incorrectly classified samples are updated as follows:

$$W_i' = W_i * e^{AoS},$$

where W_i is the sample weight of the ith sample, and W_i' is the corresponding new sample weight. The sample weights for samples that were correctly classified are updated using the following formula:

$$W_i' = W_i * e^{-AoS}.$$

Finally, the sample weights are normalized, such that they add up to 1 again:

$$W_i'' = \frac{W_i'}{\sum_{j=1}^{n} W_j'},$$

where W_i'' is the normalized W_i'.

Then the next stump is generated. This stump either utilizes weighted Gini values (based on the sample weights) or a resampled dataset. The new dataset considered for this stump would then be the same size as the original dataset, but with each sample generated by randomly sampling the original dataset (with replacement), taking into account the sample weights. As a result, samples with larger weights will occur more frequently in the generated dataset, and samples with small weights are less likely to occur at all.

This process is then repeated, generating some specified number of stumps, each with a different AoS. In order to make a classification based on these stumps, for each label, a weighted sum is computed of all stumps that classify the sample as that label. The greatest weighted sum determines the model's final classification. For a more formal description, see algorithm 3.

Algorithm 3 AdaBoost training

Require: $X \in \mathbb{R}^{n \times m}$, where $n \in \mathbb{N}$ is the number of samples, $m \in \mathbb{N}$ is the number of features, and X_i^j refers the the jth feature of the ith sample, $y \in \mathbb{R}^m$, where $m \in \mathbb{N}$ is the number of features, and y^j refers to the ith sample, $k \in \mathbb{N}$

function ADABOOST(X, y, k)
  $w_i \leftarrow 1/n$
  $trees \leftarrow \varnothing$
  for $i \leftarrow 1$ to k **do**
    $T \leftarrow Stump(X, y, w)$
    $\epsilon \leftarrow Pr(T(X) \neq y)$
    $a \leftarrow \frac{1}{2} ln(\frac{1-\epsilon}{\epsilon})$
    for $j \leftarrow 1$ to n **do**
      if $T(X_j) = y_j$ **then**
        $w_j \leftarrow w_j * exp(-a)$
      else
        $w_j \leftarrow w_j * exp(a)$
    $trees \leftarrow trees \cup \{(T, a)\}$
  return $trees$

Gradient-boosting decision trees

Gradient-boosting is another technique that utilizes decision trees (Mason et al., 1999). Although it is similar to AdaBoost, there are once again a few key differences. We will first describe the algorithm in the context of regression and then in

the context of classification.

For regression, gradient boosting starts with a single leaf instead of a stump. This leaf is equal to the average of all sample targets. If we were to stop here, the model would simply predict this average for all samples. Then, similarly to AdaBoost, another tree is generated based on the errors made by the first tree. Specifically, for each of the samples in the dataset, a pseudo residual is computed. That is, we compute the difference between the true target value of each sample and the predicted value (at this point, just the mean). We then construct a tree attempting to predict this residual error. Just like AdaBoost, the tree generated is restricted in size, but not necessarily to a stump. The output of each leaf in this tree will be the average of all residuals of the samples in that leaf. The output of the model is now the initial leaf node (the average) plus the output of this new tree. In order to prevent overfitting on the training data, the output of this second tree—and of any subsequent trees—is multiplied by a fixed value: the learning rate. This process of generating trees to predict residuals is repeated until the desired number of trees is reached. Alternatively, trees could be generated until the performance on some external validation set ceases to improve with additional trees.

Gradient boosting decision trees can also be used for binary classification, with some slight alterations. Firstly, instead of the first leaf being the average of all labels, it is equal to the log of the odds (logit): $\log(odds) = \log(\frac{N_P}{N_N})$ (where N_P is the number of positive samples and N_N is the number of negative samples). Then, the probability is calculated from this logit as follows:

$$p = \frac{e^{\log(odds)}}{1 + e^{\log(odds)}},$$

where p is the probability corresponding to the logit $\log(odds)$. Based on this probability, the pseudo residuals can then once again be computed as the difference between the sample label and the first leaf's probability. These pseudo residuals are then used to build another (constrained) tree. However, unlike with regression, instead of simply taking the average of all samples in a leaf, a more complex computation must be applied (as the first leaf's value is a logit, but the residuals were computed against the logit's corresponding probability):

$$L_i = \frac{\sum_{r \in R_i} r}{\sum_{r \in R_i} prev_r * (1 - prev_r)},$$

where L_i is the ith leaf's value, R_i is the set of residuals in leaf i, and $prev_r$ is the previously predicted probability corresponding to residual r. We can now compute the new log of odds for each sample in the dataset by summing the first leaf we generated and the first larger tree we generated, the second once again being scaled by some learning rate. Based on this new set of logits, we can compute the corresponding probabilities and pseudo-residuals, and then repeat the process to generate the desired number of trees. For a more formal description, see algorithm 4.

1.2 Shapley Values

A common issue in machine learning (ML) is that of interpretability or explainability. ML models, especially neural networks, often lack any clear explanation as to why they generate a certain prediction. We might therefore consider using simpler

Algorithm 4 Gradient Boosting algorithm

Require: $X \in \mathbb{R}^{n \times m}$, where $n \in \mathbb{N}$ is the number of samples, $m \in \mathbb{N}$ is the number of features, and X_i^j refers the the jth feature of the ith sample, $y \in \mathbb{R}^m$, where $m \in \mathbb{N}$ is the number of features, and y_j refers to the ith sample, $k \in \mathbb{N}$

function GRADIENTBOOST(X, y, k)
  $F_0(x) \leftarrow \arg\min_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$
  for $i \leftarrow 1$ to k **do**
    $r_{im} \leftarrow [\frac{\partial L(y_i, F(X_i))}{\partial F(X_i)}]_{F(x)=F_{m-1}(x)}$, for $i = 1, \ldots, n$
    $tree \leftarrow DecisionTree(X, r_{im})$
    $R_{jm} \leftarrow$ the leaves of $tree$, where j refers to the jth leaf
    $J_m \leftarrow$ number of leaves in $tree$
    for $j \leftarrow 1$ to J_m **do**
      $\gamma_{jm} \leftarrow \arg\min_\gamma \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$
    $F_m(x) \leftarrow F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
  return F_k

models, such as lone decision trees, which can be understood more intuitively. However, the performance of such models is often unsatisfactory, rendering them useless in practice.

In order to remedy this issue of interpretability, many solutions have been suggested. One such solution is that of Shapley values, a concept hailing from game theory (Shapley, 2016). In its adaptation to machine learning, we will consider individual features of a dataset (age, sex, etc.) to be 'players', the final prediction of the model to be the 'total payout', the process of generating a prediction a 'game', and the effect of a single feature on the output (in terms of deviating from the mean) to be the 'gain' (Lundberg and Lee, 2017). The Shapley value is then defined at the "average marginal contribution of a feature value across all possible coalitions" (Molnar, 2020). More specifically, for any feature set $F \in \mathbb{R}^n$, we can compute the Shapley value ϕ_f of a feature $f \in F$ by computing the effect (in terms of model prediction) of adding f to a pre-existing feature coalition $C \subset F$ (where $f \notin C$). We then repeat this process for all possible coalitions $C \subset (F \backslash \{f\})$. The average of all the contributions for all possible coalitions then gives us the Shapley value for feature f.

This computation presents two main issues. Firstly, since most machine learning models do not natively accept missing values, we have to simulate the missingness to compute the contributions of a feature to all possible coalitions. We can do this by resampling the features that we intend to be missing from the dataset repeatedly, and averaging the contributions computed for these different missing value replacements. Secondly, as the number of features increases, the computation time for these Shapley values increases exponentially (as the number of possible coalitions increases exponentially). One way to remedy this would be to only use a smaller set of samples from the set of all possible coalitions.

1.2.1 Interpretation, advantages, and disadvantages

The Shapley value ϕ_f for a feature f can be interpreted as how much the feature contributed to the prediction of a particular sample, compared to the average prediction for the dataset. Figure 1.6 shows a plot of Shapley values for a random forest trained to predict cervical cancer for a particular woman within this dataset. Figure

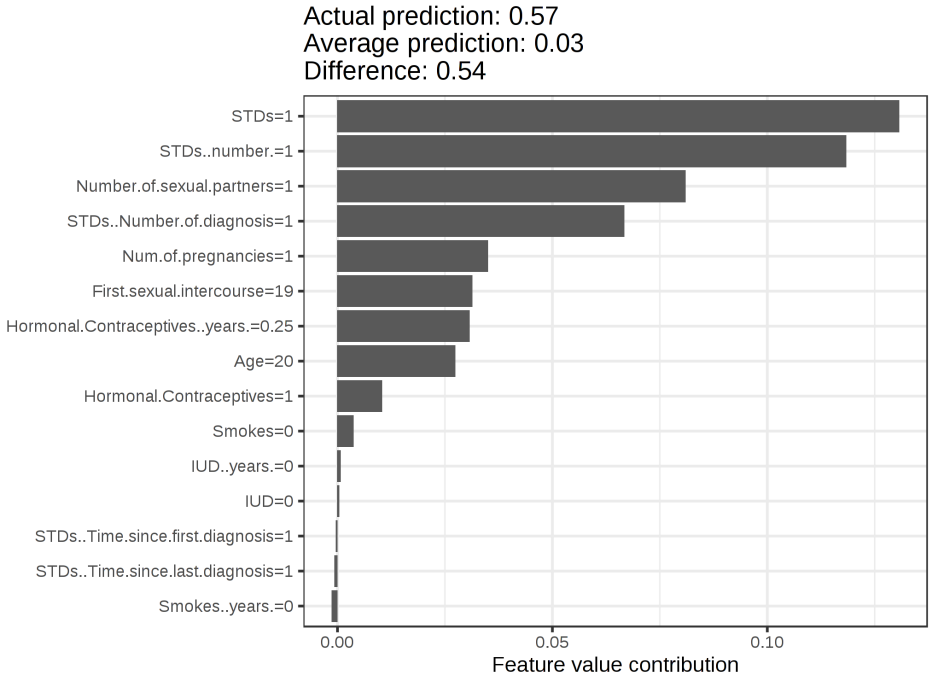


FIGURE 1.6: Adapted from Molnar, 2020, "Shapley values for a woman in the cervical cancer dataset. With a prediction of 0.57, this woman's cancer probability is 0.54 above the average prediction of 0.03. The number of diagnosed STDs increased the probability the most. The sum of contributions yields the difference between actual and average prediction (0.54). "

1.7 shows Shapley values for a random forest trained to predict the number of bikes rented on a particular day.

Shapley values adhere to four key properties: Efficiency, Symmetry, Dummy, and Additivity. These are defined as follows (Molnar, 2020):

Efficiency: "The feature contributions must add up to the difference of prediction for x and the average"

$$\sum_{j=1}^{p} \phi_j = \hat{f(x)} - E_X(\hat{f(X)})$$

Symmetry: "The contributions of two feature values j and k should be the same if they contribute equally to all possible coalitions. If"

$$val(S \cup \{x_j\}) = val(S \cup \{x_k\})$$

for all

$$S \subseteq \{x_1, \ldots, x_p\} \backslash \{x_j, x_k\}$$

then

$$\phi_j = \phi_k.$$

Dummy: "A feature j that does not change the predicted value – regardless of which coalition of feature values it is added to – should have a Shapley value of 0. If"

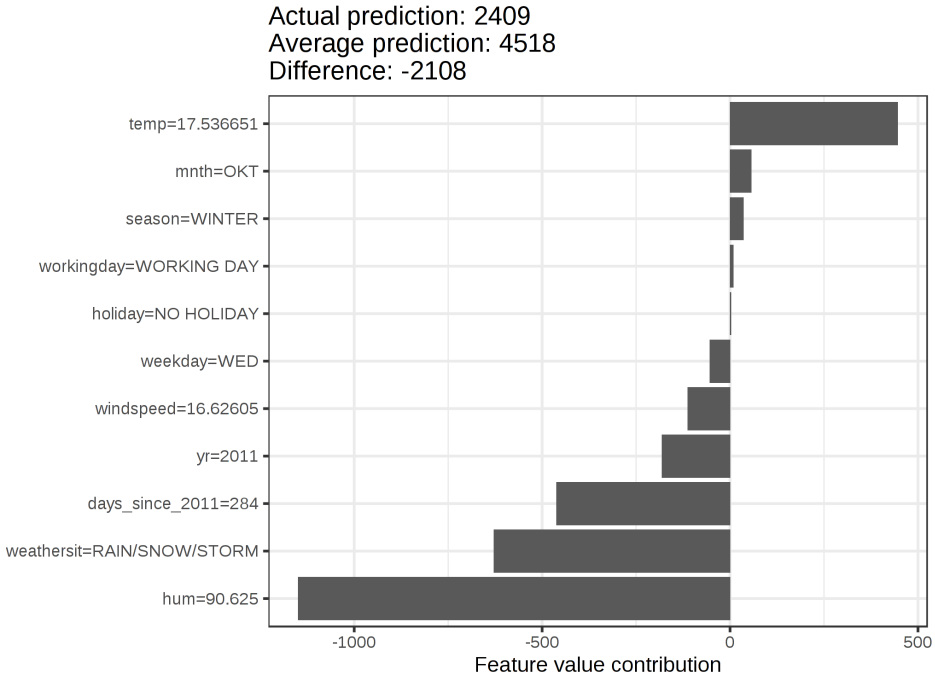


FIGURE 1.7: Adapted from Molnar, 2020, "Shapley values for day 285. With a predicted 2409 rental bikes, this day is -2108 below the average prediction of 4518. The weather situation and humidity had the largest negative contributions. The temperature on this day had a positive contribution. The sum of Shapley values yields the difference of actual and average prediction (-2108). "

$$val(S \cup \{x_j\}) = val(S)$$

for all

$$S \subseteq x_1, \ldots, x_p$$

then

$$\phi_j = 0$$

Additivity: "For a game with combined payouts $val + val^+$ the respective Shapley values are as follows:"

$$\phi_j + \phi_j^+.$$

Since random forests are just a collection of decision trees—with the prediction being the average of all these trees—the additivity property ensures we need only compute Shapley values for all individual trees and average them.

In addition, because of the efficiency property of Shapley values, this approach can guarantee a fair distribution of the difference between a prediction and the average prediction, allowing Shapley values to give a full explanation. By 'fair', we mean that equally contributing features have equal Shapley values. Moreover, since Shapley values satisfy the four previously mentioned axioms, this gives the explanation a proper mathematical foundation, as opposed to most other methods.

However, there are also some potential issues with Shapley values. Most importantly, Shapley values require a lot of computation time. This means that, in real-world applications, exact Shapley values would not be feasible. Thus, approximations have to be made. This can be done in two ways. The first approach is to reduce the number of coalitions that are evaluated for each Shapley value. The second approach is to decrease the number of samples M drawn to simulate the missingness of certain features. Decreasing M reduces the computation time, but also results in increased variance in the computed Shapley value.

Using these Shapley values, Lundberg et al., 2020a developed a framework for interpreting model predictions, which they called SHAP (SHapley Additive exPlenations). In this framework, they also implemented an algorithm to compute exact SHAP values of tree ensemble models with high performance in practice. They also published their implementation as the Python package SHAP, which we utilized for this thesis.

1.3 Hyperparameter optimization

All of the previously described machine learning algorithms have a varying number of different hyperparameters. These are parameters that are not estimated during the fitting of the model, but are rather parameters that control the model architecture or the way in which regular parameters are estimated. Normally, one might manually tweak these hyperparameters, but there are also various techniques to estimate these parameters automatically.

Firstly, one of the simplest hyperparameter optimization strategies is grid search. That is, for each hyperparameter $h \in H$, we have some set of values V_h. In grid search, we simply evaluate each element $V \in \times_{h \in H} V_h$ using some loss function $L(V)$ and return the best parameter set V. While this will result in the most optimal solution, at least when the parameter space is discrete, this is generally not a feasible strategy for real-world problems unless the hyperparameter space is very low-dimensional.

Another approach, therefore, is that of random search. Instead of enumerating over all possible hyperparameter combinations, we simply define a distribution $D(h)$ for each hyperparameter $h \in H$. We then repeatedly sample these distributions to obtain a hyperparameter set which we can again evaluate using some loss function $L(V)$. We repeat this process some $n \in \mathbb{N}$ times and return the best hyperparameter set V. While this approach works a lot better than simple grid search (Bergstra and Bengio, 2012), it might still prove problematic in situations where the computation of the loss $L(V)$ takes a long time or when the hyperparameter space is very high-dimensional.

A third approach, therefore, is that of Tree-structured Parzen Estimators (TPE, Bergstra et al., 2011). TPE is a sequential optimization technique (as opposed to grid and random search), meaning it determines what parameter set to try next based on previously obtained results. That is, TPE builds a probability model of the objective function and uses this model to select the most promising hyperparameters to evaluate next.

1.4 Missing data

In real-world datasets, it is common for certain features to be missing for certain samples. In a clinical context, this can be the result of doctors simply choosing not to

measure values for certain patients or due to failing tests. More formally, there are three kinds of missing data (Rubin, 1976):

1. **Missing completely at random (MCAR)**: Values in a dataset are missing completely at random if the cause of their absence is independent of both observable variables and unobservable parameters of interest.

2. **Missing at random (MAR)**: This occurs when data is not missing at random, but where the missingness can be fully accounted for by other variables.

3. **Missing not at random (MNAR)**: In this case, the missingness of the variable is related to its value (e.g. smokers who do not answer the question asking whether they are smokers, because they are smokers).

This presents multiple problems. Firstly, some (implementations of) machine learning models do not support missing values in the input data, thus they cannot be directly used with these models. In order to resolve this, one could simply fill in the mean of a feature for all its missing values, allowing the data to be passed to the model. While this does indeed allow the model to operate, it might not result in the most optimal performance, as this operation disturbs the distributions in several ways; it will underestimate the variance, disturb relations between variables, and bias any estimate (except for the mean, if the data are MCAR) (Van Buuren, 2018).

Another approach might be to simply drop any rows/columns in the data that contain missing values. However, throwing out data can be undesirable as well, especially if the number of missing values is large, meaning the majority, if not all, data would end up being dropped. Thus, it is often desirable to impute the data in some way. One common way of doing this is simply to take the mean, median, or mode of all present values of a certain feature, and use this value to impute the missing values of that feature (Donders et al., 2006), though this will likely still result in imputations that fail to approximate the true values.

Yet another, more sophisticated approach is that of k-nearest neighbour imputation (Troyanskaya et al., 2001), which uses the present values of the k nearest neighbours of a sample to estimate the missing value. This is done either by averaging the neighbours' values uniformly or by means of a weighted average based on the distances between the neighbours and the sample.

Finally, one even more sophisticated approach is Multivariate Imputation by Chained Equations (MICE) (Buuren and Groothuis-Oudshoorn, 2010). This approach models each feature with missing values as a function of other features, and uses that to estimate the missing values and impute them. It does this in an iterative round-robin fashion. At each iteration, a feature is designated as the target y, and the other features are treated as inputs X. For all known values in y, a regressor is then fit on X. This is iteratively done for each feature and is then repeated for some user-specified n rounds. The results of the final imputation round are used as the output imputations of the approach. Unlike the previous approaches, this method can also generate multiple imputations for the same missing value, allowing for a better representation of the underlying distribution.

However, one final issue regarding missing data—which was not specifically considered in any of the previous approaches—is that the patterns of missing data (also referred to as simply missingness) for a sample may be indicative of its classification. A model that operates directly on non-imputed data, or that operates on data that was only imputed using basic imputation methods (such that the underlying missingness patterns are still deducible) may overfit on these patterns. While,

FIGURE 1.8: A CT scanner.

in some circumstances, these patterns may generalize to new, unseen data, this need not be the case for all problem domains. As such, the extent to which a method masks the missingness of values may also be a relevant factor in deciding how to deal with missing data.

1.5 Computerized Tomography

Computerized tomography (CT) allows one to generate three-dimensional images of some subject using repeated x-ray images (Cnudde and Boone, 2013). More specifically, a CT scanner operates by rotating an x-ray source around the subject within a doughnut-shaped structure, which is referred to as a 'gantry". During a CT scan, the patient moves through the gantry, whilst the rotating x-ray source shoots narrow x-ray beams at and through the subject. Figure 1.8 shows an example of a CT scanner. Directly opposite to the x-ray source are digital x-ray sensors, which pick up the signal that passes through the subject and send them to a computer. For each revolution of the x-ray source, the computer constructs a two-dimensional 'slice' of the subject, with a slice thickness somewhere in the range of 1-10 millimetres. After completing a revolution, the motorized bed on which the subject is situated is moved slightly through the gantry and another revolution is initiated. This process is continued until the desired number of slices (and thus revolutions) is completed. These slices can then be stacked into a three-dimensional image.

These three-dimensional images can show bones, organs, and tissues and can be viewed either in 3D or by moving through individual slices. They can help to identify tumors or lesions within the abdomen and to identify heart disease or abnormalities, among many more applications. Particularly relevant to this thesis is their application in detecting diseases affecting the lungs, which includes COVID-19. Figures 1.9 and 1.10 show examples of (2D projections of) chest CT scans.

1.6 SARS-CoV-2

Coronaviruses are a group of viruses infecting both humans and other animals. In 2002, a highly pathogenic coronavirus of zoonotic origin emerged in humans, named severe acute respiratory syndrome coronavirus (SARS-CoV), causing fatal respiratory illness. Then, in 2012, another coronavirus emerged, namely the Middle East respiratory coronavirus (MERS-CoV), which also causes fatal respiratory illness, marking coronaviruses as a significant health concern for the twenty-first century (Hu et al., 2020).

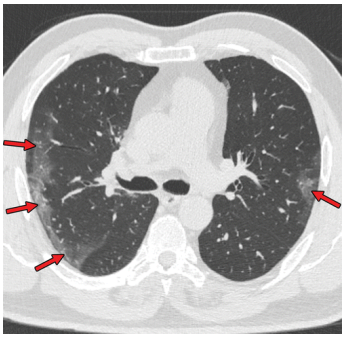
In late December 2019, several health facilities in Wuhan, China, reported clusters of cases with pneumonia of unknown origin. Most of these cases were linked to the Huanan Seafood Wholesale Market, which sells, among other things, live animals (Zhu et al., 2020; Gralinski and Menachery, 2020). Once again, a coronavirus of zoonotic origin had emerged in humans. The virus was designated SARS-CoV-2, and the disease it brings about was designated 'coronavirus disease 2019' (COVID-19). The virus, being highly transmissible, quickly spread across the globe, eclipsing both SARS-CoV and MERS-CoV in terms of both the total number of infected humans and spatial extent. Now, more than a year since the WHO declared COVID-19 a pandemic, the virus still continues to put great pressure on economies and healthcare infrastructure across the world.

1.6.1 Diagnosis

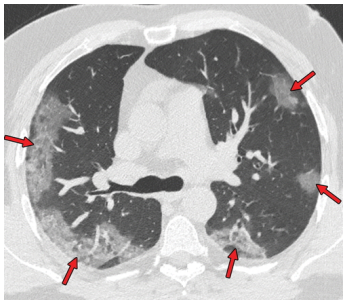
In order to prevent the spread of the virus, it is imperative that the disease is detected in an early stage. Currently, molecular testing of SARS-CoV-2 nucleic acid is the gold standard. Many such tests are commercially available, with detection times ranging between several minutes to hours, depending on which particular technology is used (Chan et al., 2020; Konrad et al., 2020; Lu et al., 2020). While this molecular detection generally performs quite well, there are a few factors that may hamper it. Firstly, while SARS-CoV-2 can be detected from a number of respiratory sources (e.g. throat swabs, posterior oropharyngeal saliva, nasopharyngeal swabs, sputum and bronchial fluid), the viral load is most present in lower respiratory tract samples (Zhou et al., 2020; Zou et al., 2020; Pan et al., 2020; To et al., 2020; Wang et al., 2020; Han et al., 2020), and viral load may also already drop from its peak level on disease onset (Zou et al., 2020; Wölfel et al., 2020). Consequently, false negatives can be relatively common—Arevalo-Rodriguez et al., 2020 found that up to 54% of COVID-19 patients may have an initial false-negative PCR outcome–and multiple detection methods should be considered for COVID-19 diagnosis.

As such, other detection methods have been utilized, one of which is chest CT. Chest CT scans showed typical features in COVID-19 patients, including bilateral multilobar ground-glass opacities with a peripheral or posterior distribution (Kwee and Kwee, 2020). Figures 1.9a and 1.9b show examples of COVID-19 pneumonia in chest CT scans of patients. Figure 1.10 shows additional chest CT abnormalities found in patients. These three figures demonstrate some of the more common ways COVID-19 presents itself in chest CT scans (>70%), but there are additional abnormalities indicative of COVID-19 that can present themselves.

These CT scans can be graded with the COVID-19 Reporting and Data System (CO-RADS), which assigns a value of 1-5, indicating the level of suspicion of COVID-19 infection based on chest CT scans (Prokop et al., 2020). Table 1.2 shows the different CO-RADS levels and what they indicate approximately. Additionally, CO-RADS 6 indicates a positive PCR test.



(A) A 59-year old male patient with positive RT-PCR SARS-CoV-2 test results



(B) A 47-year male patient old with positive RT-PCR SARS-CoV-2 test results

FIGURE 1.9: COVID-19 pneumonia with typical imaging features, axial nonenhanced chest CT images (lung window) (adapted from Kwee and Kwee, 2020)

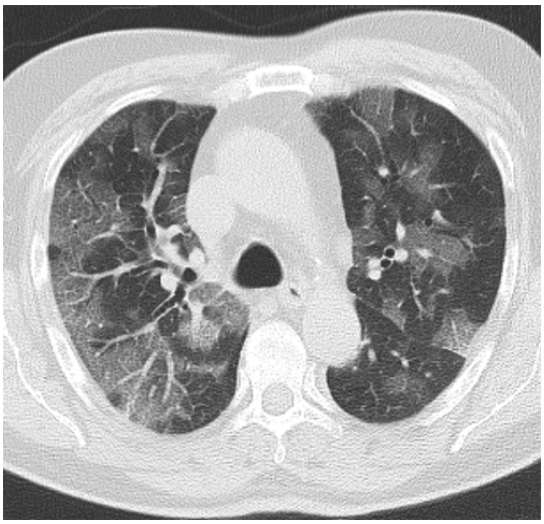


FIGURE 1.10: Adapted from Kwee and Kwee, 2020, "Chest CT abnormalities of relatively high prevalence in COVID-19. Axial nonenhanced chest CT image (lung window) shows bilateral ground-glass opacities and dilated segmental and subsegmental vessels, mainly on the right, in a 70-year-old man with positive RT-PCR test results for SARS-CoV-2."

	Level of suspicion COVID-19 infection	**CT findings**
CO-RADS 1	None	Normal or non-infectious abnormalities
CO-RADS 2	Low	Abnormalities consistent with infections other than COVID-19
CO-RADS 3	Intermediate	Unclear whether COVID-19 is present
CO-RADS 4	High	Abnormalities suspicious for COVID-19
CO-RADS 5	Very high	Typical COVID-19
CO-RADS 6	Positive PCR	

TABLE 1.2: CO-RADS levels (adapted from Prokop et al., 2020)

1.7 CO-RADS-AI

This thesis builds upon the results of the CO-RADS AI study by Lessmann et al., 2020. This AI system consists of three consecutive deep learning systems that attempt to assign CO-RADS scores to CT scans. The first of these systems is a lobe segmentation and labelling network, implemented using a relational two-stage U-net architecture and pretrained on 4000 chest CT scans from the Genetic Epidemiology of Chronic Obstructive Pulmonary Disease study. This model was then fine-tuned to 400 scans from the CO-RADS AI study.

The second deep learning system was used to predict CT severity scores. This was done using a three-dimensional U-Net (Ronneberger, Fischer, and Brox, 2015) with 108 CT scans and corresponding reference delineations to segment ground-glass opacities and consolidation in the lungs. The actual CT severity score was then derived from these segmentation results by computing the percentage of affected parenchymal tissue per lobe.

Finally, for CO-RADS score prediction, a three-dimensional inflated inception architecture was used (Carreira and Zisserman, 2017). This model was pretrained on ImageNet and Kinetics, and was then trained on 368 CT scans from the RadboudCOVID dataset. The model uses the CT image as input, together with areas of abnormal parenchymal lung tissue detected by the severity scoring algorithm.

Chapter 2

Data and methods

The following sections describe the datasets and methods we employed in this thesis. The first few sections discuss the datasets, the models and specific machine learning techniques we employed, adapted, or developed. The final section discusses the experimental setup and describes how we evaluated all of the approaches in the preceding sections.

2.1 Datasets

For this thesis, three datasets of potential COVID-19 patients were used. Each of these datasets contains CT scans as well as lab values and other patient data such as sex and age. The next two sections will provide an overview of the structure of these datasets.

2.1.1 RadboudCOVID

The RadboudCOVID dataset contains at least one lab value for 447 patients. Some of these patients were involved in multiple studies and, as such, can be considered as separate samples. Patient data were grouped based on how close in time they were measured to their related CT scans. Clinical values within three days before and after the CT scan were assigned to the corresponding study, and PCR tests within 14 days before and after the scan were assigned to the corresponding study. Based on these groupings, there are 514 samples that have at least one lab value. The dataset contains 796 chest CT scans. Again, some of these scans were performed on the same patient (but with a significant amount of time between the scans). In total, there are 813 samples, of which 497 samples have both CT data and lab values. All samples have CO-RADS values associated with them, and 155 samples have PCR values associated with them. This dataset will henceforth be referred to as simply the RUMC dataset. Figure 2.1 shows the missingness present in this data.

2.1.2 RUMC/CWZ

The RUMC/CWZ dataset contains data from both the RadboudUMC hospital and the Canisius Wilhelmina Ziekenhuis (CWZ) hospital. It contains 434 different patients and 26 clinical features. 423 patients have chest CT scans associated with them, and each patient has an associated binary diagnosis value. This dataset will henceforth be referred to as simply the CWZ dataset. Figure 2.2 shows the missingness present in this data.

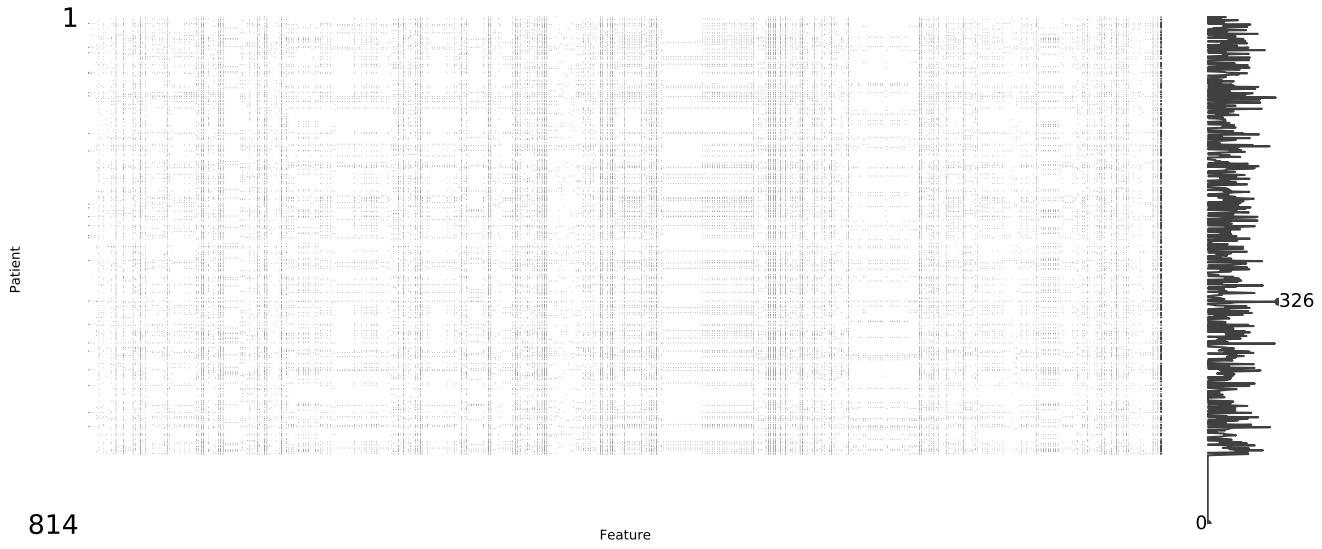


FIGURE 2.1: Matrix visualization of the missingness of clinical features in the RUMC dataset. Black cells indicate that the value is present, white cells indicate that the value is missing. The graph on the right shows the total number of values present for a particular patient. Finally, the two numbers on the right indicate the smallest and largest number of features present in any patient.

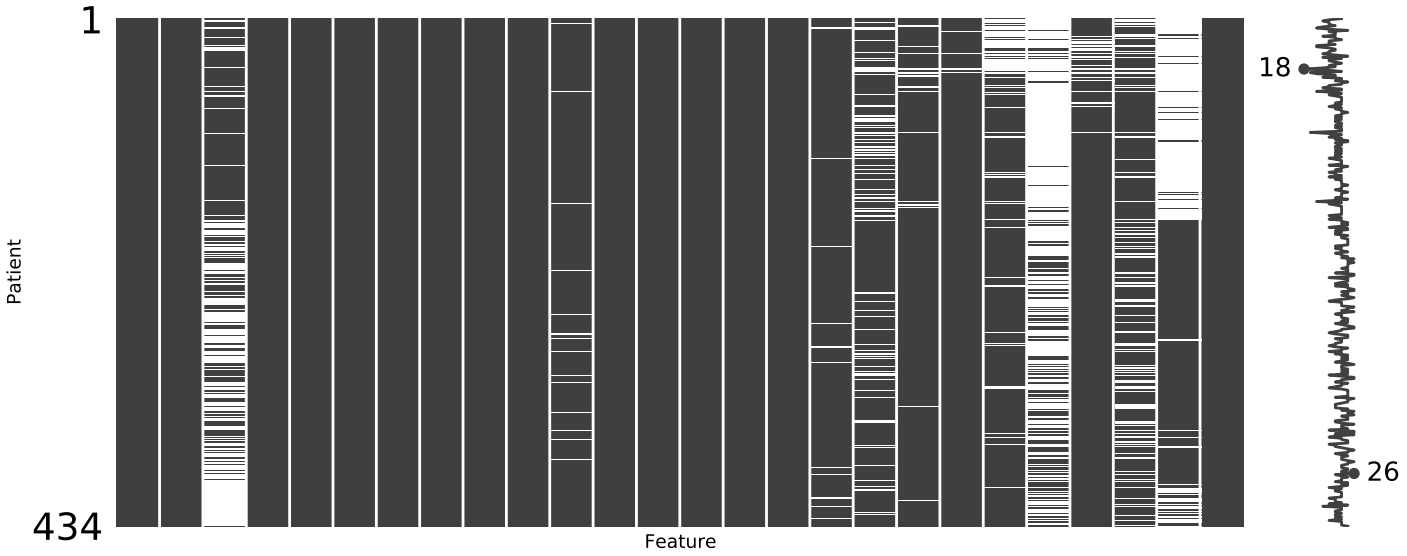


FIGURE 2.2: Matrix visualization of the missingness of clinical features in the CWZ dataset. Black cells indicate that the value is present, white cells indicate that the value is missing. The graph on the right shows the total number of values present for a particular patient. Finally, the two numbers on the right indicate the smallest and largest number of features present in any patient.

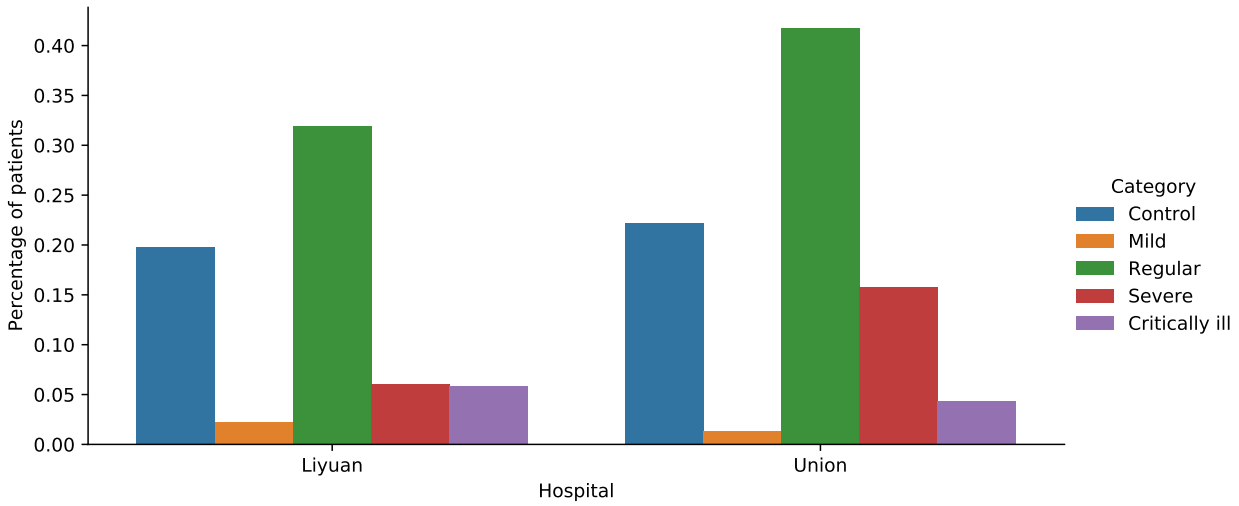


FIGURE 2.3: Comparison of patient morbidities between the Union and Liyuan hospitals in the iCTCF dataset.

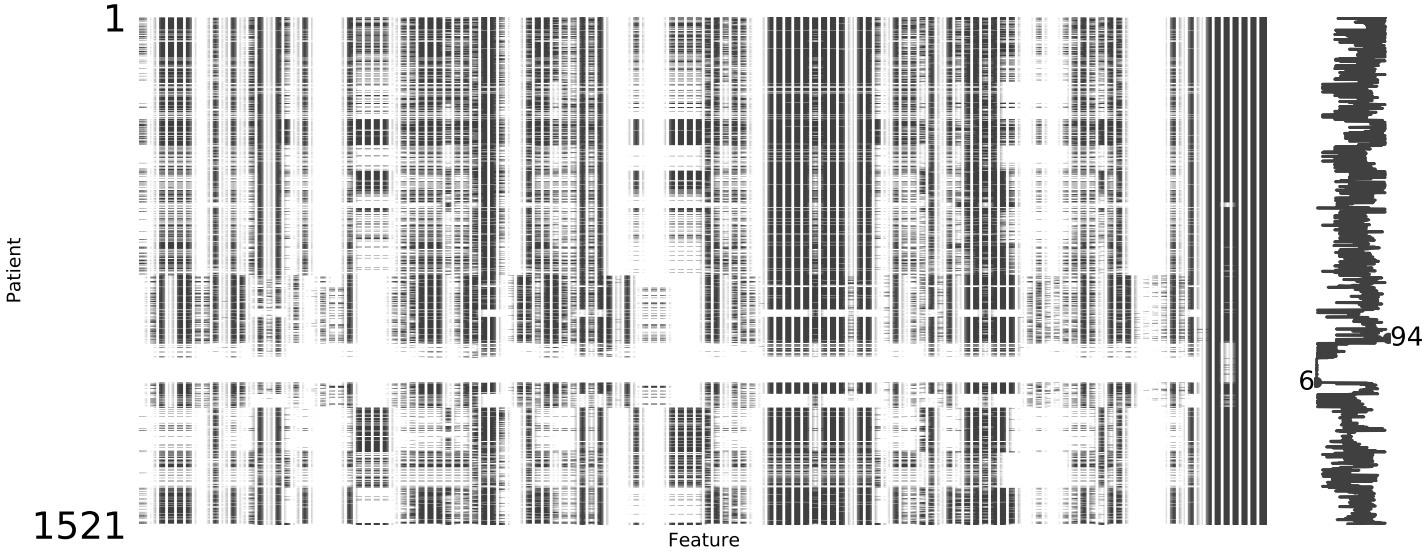


FIGURE 2.4: Matrix visualization of the missingness of clinical features in the iCTCF dataset. Black cells indicate that the value is present, white cells indicate that the value is missing. The graph on the right shows the total number of values present for a particular patient. Finally, the two numbers on the right indicate the smallest and largest number of features present in any patient.

2.1.3 iCTCF

The integrative CT images and CFs for COVID-19 (iCTCF) dataset contains a total of 1521 patients (Ning et al., 2020). The dataset contains a total of 130 different types of clinical features. The dataset is divided into two cohorts: one consisting of 1170 patients and the other of 351 patients. Each patient's sample is marked as either negative (Control), mild/regular (Type I), or severe/critically ill (Type II). 1084 of these samples have chest CT scans associated with them. Furthermore, each sample is also marked with mortality (i.e., deceased or cured) and PCR outcome (945 were positive). The data were obtained from two separate hospitals: the Union hospital (HUST-UH, 1128 patients) and the Liyuan hospital (HUST-LH, 395 patients). Figure 2.3 shows the distributions of patients in the categories described above for both hospitals. Suspected patients were not considered in this thesis, as their target labels were uncertain. Figure 2.4 shows the missingness present in this data. Ning et al., 2020 also developed predictive models based on these data and we will thus also compare our results against theirs.

2.2 Visual feature extraction

Since we wished to make predictions based on both CT images and clinical features, we needed a method to extract visual features from chest CT scans. To this end, we adapted the CNN by Lessmann et al., 2020, described in section 1.7. Specifically, we used the CORADS-AI system— pretained on the RadboudCOVID dataset—and ran this on each CT image, extracting all values from the last layer of the CNN. Finally, we concatenated the generated visual feature vector with the clinical data.

2.3 COVID-19 prediction based on visual and clinical features

After transforming the CT images into high-level feature vectors, we could then train a model to make predictions based on the combined clinical/visual features.

2.3.1 Estimators

We considered a few different types of models:

1. Logistic Regression (LR)
2. Gradient Boosting Decision Trees (GBDT)
3. Bagged Gradient Boosting Decision Tree (BaggedGBDT)
4. Scikit-learn Random Forest (RFsklearn)

For the LR and RFsklearn models, we used the implementations provided by the scikit-learn Python package (Pedregosa et al., 2011). For the GBDT model, we used the implementation provided by the LightGBM Python package (Ke et al., 2017). For the BaggedGBDT model, we used scikit-learn for the bagging functionality and LightGBM for the underlying gradient boosting model.

Preprocessing

For all of the models and datasets, only features with at least 10% of its values present were considered. Additionally, LR and RFsklearn used mean imputation for all experiments, except for those that involved some other imputation method, as neither model accepts missing data directly. Finally, for LR, we standardized all data by removing the mean and scaling to unit variance.

2.3.2 Hyperparameter optimization

For the LR, GBDT, and RFsklearn models, we employed hyperparameter optimization using TPE. We did not use it for the BaggedGBDT model, as training each individual instance of this model was too computationally intensive for hyperparameter optimization. The objective function we used for each of these hyperparameter optimization strategies was the average Area Under the Curve (AUC) of all splits of 5-times repeated 5-fold cross-validation.

2.4 Quantifying overfitting on missingness

One issue we encountered with some of the datasets we looked at was the large number of missing values. In general, missing values were the result of humans deciding not to measure these values. Thus, one might expect the absence of these values to be indicative of the sample's label. Consequently, this means that a model could fit on these patterns of missingness rather than the actual values that were measured. This behaviour might be considered undesirable and, as such, we employed various strategies to quantify the extent to which a model (might) (over)fit on patterns of missingness.

2.4.1 SHAP values

Since SHAP values provide a numerical value of the contribution of a feature in a particular sample on a model's prediction, we can utilize these values to determine to what extent a missing value affects the output. If a missing value's associated SHAP value is 0, then we would assume that it does not affect the model's output. To compute SHAP values in this thesis, we used the SHAP Python package (Lundberg et al., 2020a; Lundberg et al., 2020b).

One caveat, however, occurs when the data were imputed before being passed to the model. In this case, we would not necessarily expect the SHAP values to be (close to) 0. This can be understood with the following hypothetical. Suppose one had some method to impute data in such a way as to fill in the values that one would have obtained had they actually been measured. Since these values would then be perfectly accurate, there would be no reason for a model not to rely on them. Thus, we would not expect—or even want—SHAP values for such features to be zero. As a result, other methods are necessary for such cases.

2.4.2 Fitting on missingness indicators

Another approach we utilized was to fit ML models on missingness indicators. By missingness indicators, we mean Boolean values that indicate whether a certain value was missing in the original dataset. By training an ML model on these missingness indicators (and, crucially, not any of the original numerical and categorical

data), we can obtain an indication as to how indicative the patterns of missingness are of the target label. If we obtain high performance on this task, we know that they are very indicative of the target label, though we cannot say if any model trained on the original data would necessarily owe their performance to the missingness patterns. If the performance on this task is bad, it suggests that the patterns are not very indicative, and we also know that any model trained on the original dataset would not owe much of its performance to this information.

2.4.3 Predicting missingness in imputed data

While the approach in the previous section can tell us whether missingness is indicative of the target label in a dataset, it cannot tell us to what extent it can extract this information, especially when the data were imputed before being passed to the model.

We evaluated the extent to which a model is able to extract missingness from imputed data by simply training it to predict whether values in the input were missing or not, rather than the normal target label. This can then give an indication both as to how good an imputer is at obscuring missingness, as well as how good a model is at extracting it. Note that, since we assume most values not to be missing completely at random, we would expect it to be possible to predict missingness even in the extreme case of a perfect imputer (that is, an imputer that fills in the values that would have been measured, had they been measured). Thus, we would not expect an imputation method to exist that could drive all models to same-as-random performance.

2.4.4 Cross-hospital prediction

One final way to evaluate the extent to which missingness affects models is by predicting on some dataset with different missingness patterns. In this thesis, we will achieve this by means of cross-hospital prediction, as we believe there is likely to be some differences in the exact methodologies used to determine which values to measure between hospitals. However, there may be more differences between data collected in different hospitals, which means reductions in performance may be the result of those effects rather than (just) the effects of differing missingness patterns.

2.5 Preventing overfitting on missingness

The previous section discussed ways in which to quantify overfitting on missingness. This section will consider various methods for preventing this phenomenon.

2.5.1 Imputation

The most common way of dealing with missing values is to impute them. This is especially so as many models do not accept missing values whatsoever, in which case the missing values must be dealt with beforehand. We tested the following imputation methods:

1. Mean imputation
2. MICE imputation using Bayesian Ridge regression
3. k-nearest neighbour imputation with $k = 5$

While these methods might prevent models such as GBDT from fitting directly on missingness (as they can natively handle missing values, thus do not require imputation), it presents some difficulty in ascertaining the extent to which the imputation method is actually succeeding at this, as mentioned in section 2.4.1. While the methods described in sections 2.4.2, 2.4.3, and 2.4.4 can provide some insight into this, they cannot provide conclusive answers, rendering this method somewhat problematic.

2.5.2 SHAP zeroing

Another approach we used is that of SHAP zeroing. Since SHAP values provide a breakdown of what features contributed to the final prediction and by how much, we can compute the prediction by simply summing all of these values, plus the mean prediction. Furthermore, we could also remove all SHAP values associated with missing data, resulting in a prediction made based on only the present values.

2.5.3 Ensembled multiple imputation

Another approach we considered was multiple imputation. Specifically, we considered an approach of per feature imputation, where each missing value is randomly imputed by sampling from all present values for that feature.

The procedure we used is as follows. Firstly, we repeat the dataset k times (that is, if we had n samples in the dataset previously, the new dataset will have $n * k$ samples). Next, for each repetition of the data, we randomly impute the data by filling in values sampled from all values present for each feature. We then train a homogeneous ensemble of models on this new repeated dataset, where each model is trained on one repetition of the data.

Since we want to obtain a single prediction for each sample at test-time, we considered two different approaches. The first approach we implemented is to use the same multiple imputation strategy used during training at test-time, run each model in the ensemble on each new randomly imputed dataset, and then average the ensemble's predictions.

The second approach we implemented is to only impute the data once, using simple mean imputation. Then, each model in the ensemble is used to generate predictions for the same test dataset, and, finally, the predictions are averaged together.

2.5.4 Native missingness handling

Finally, the last approach we tried was to develop tree-based models that can work with missing data natively. While all models implemented by LightGBM are already able to handle missing values, they simply ignore such values during a split and then later allocate them to the side that reduces the loss the most. This approach will inevitably fit on missingness. We therefore looked at the following alternatives:

For each split on a particular feature,

1. assign samples that lack this feature to the branch that would already have the largest number of instances.

2. assign samples that lack this feature to all branches, weighting these samples based on the proportion of samples that were already going to each particular branch.

3. assign samples that lack this feature to a random branch.

In the end, we settled on and evaluated the second approach. As we were unable to find any prominent Python packages implementing such an approach, we developed our own implementation of this approach.

2.6 Experimental setup

2.6.1 Toy dataset evaluation

To evaluate the extent to which the methods we considered were capable of dealing with missingness adequately, we constructed a procedure to augment simple toy machine learning data for this purpose. Specifically, we augmented it in such a way as to make its missingness predictive of the target label in the training set, but not in the test set. More formally, we used the following procedure:

1. Generate a random probability vector $miss^-$ of size n, where n is the number of features in the dataset.
2. Generate another random probability vector $miss^+ = miss^-$ and add random Gaussian noise with a mean of 0 and a standard deviation of 0.1.
3. Drop features of samples with negative labels randomly according to $miss^-$ and features of samples with positive labels randomly according to $miss^+$ in the training data.
4. Drop features of samples with negative labels randomly according to $miss^+$ and features of samples with positive labels randomly according to $miss^-$ in the test data.
5. Add random Gaussian noise to both training and test data with a mean of 0 and a standard deviation of 1.

We used the Breast Cancer Wisconsin (Diagnostic) dataset from Street, Wolberg, and Mangasarian, 1993 as an input for this procedure. To evaluate the methods described in section 2.5, we ran the procedure 100 times using random train/test splits and evaluated the methods on each. This experiment was intended to help answer research subquestion 6.

2.6.2 COVID-19 prediction in real data

For the iCTCF, RUMC, and CWZ datasets, we evaluated all base estimator models (as discussed in 2.3.1), both with mostly default hyperparameters and optimized hyperparameters as discussed in 2.3.2, with the exception of BaggedGBDT. We also ran all of these evaluations both in the case of only clinical features only and when combining clinical and visual chest CT features.

For iCTCF, the models were trained on the first cohort of the dataset and evaluated on the second, and the target label used was the PCR outcome for each patient. For RUMC, the test and validation sets defined by Lessmann et al., 2020 was used, and for this dataset the target label was also the PCR outcome for each patient. For the CWZ dataset, we used our own random split, and the target label in this case was the clinical diagnosis for each patient. All trained models were evaluated using AUC, F1 scores, precision, and recall.

We also ran these same evaluations for cross-hospital prediction. Specifically, we trained all base models on CWZ data and tested on RUMC data and vice versa for

the CWZ dataset. Similarly, for the iCTCF dataset, we trained the models on the Union hospital data and tested on the Liyuan hospital dataset and vice versa.

These experiments were intended to answer the first two research subquestions.

2.6.3 Evaluating the effects of missingness

Next, we used the methods described in section 2.4 to evaluate the extent to which missingness is fitted on and the extent to which missingness is predictive in all three datasets. This experiment was intended to answer research subquestions 4 and 5.

2.6.4 Preventing overfitting on missingness

Finally, we evaluated all of the methods in section 2.5 for both cross-hospital prediction and using the test sets described in section 2.6.2. For SHAP zeroing, we also employed hyperparameter optimization as described in section 2.3.2, but not for any of the other methods. All trained models were once again evaluated using AUC, F1 scores, precision, and recall. These experiments were intended to help answer research subquestions 4, 5 and 6.

Chapter 3

Results

	Dataset name	Test split	Training/validation split
0	CWZ	Random 20% split (excluding training data of CORADS-AI, 86 samples)	Remaining data (347 samples)
1	iCTCF	Second cohort (excluding patients marked as Suspected, 351 samples)	First cohort (excluding patients marked as Suspected, 922 samples)
2	RUMC	Predefined validation and test splits (50 samples)	Predefined training split (103)
3	CWZ: CWZ	Data from CWZ hospital (262 samples)	Data from RUMC hospital (172 samples)
4	CWZ: RUMC	Data from RUMC hospital (172 samples)	Data from CWZ hospital (262 samples)
5	iCTCF: Liyuan	Data from Liyuan hospital (excluding patients marked as Suspected, 264 samples)	Data from Union hospital (excluding patients marked as Suspected, 1009 samples)
6	iCTCF: Union	Data from Union hospital (excluding patients marked as Suspected, 1009 samples)	Data from Liyuan hospital (excluding patients marked as Suspected, 264 samples)

TABLE 3.1: The various datasets and cross-hospital versions of those datasets we evaluated, as well as the test and train splits we used for each.

The following sections discuss the results of the experiments described in section 2.6. The datasets, the cross-hospital versions of those datasets, and the train and validation splits used for these experiments are summarized in table 3.1. The full results are shown in appendix B.

3.1 Within-dataset prediction

Firstly, we evaluated all base models on each of the three datasets with mostly default hyperparameters, as well as using hyperparameters obtained by means of hyperparameter optimization with TPE. The results of these evaluations are shown in table 3.2 as the experiments 'Base' and 'Optimized', in terms of AUC. The models were also ranked based on one-sided Delong tests, which is shown in the 'Rank' column of this table. Models were considered significantly different for $p < 0.01$. It

Experiment	Dataset Model	CWZ	iCTCF	RUMC	Rank
	CORADS-AI	0.800	0.761	0.464	-
Base	GBDT	0.870; 0.836	0.940; 0.899	0.859; 0.835	1
Imputation: MEAN	GBDT	0.862; 0.836	0.925; 0.895	0.794; 0.782	1
Base	BaggedGBDT	0.869; 0.806	0.911; 0.889	0.881; 0.875	1
SHAP zeroing	GBDT	0.862; 0.841	0.908; 0.861	0.883; 0.843	2
Imputation: MEAN	BaggedGBDT	0.862; 0.804	0.881; 0.870	0.833; 0.815	2
Imputation: KNN5	GBDT	0.858; 0.844	0.823; 0.854	0.792; 0.732	3
SHAP zeroing	RFsklearn	0.876; 0.821	0.919; 0.838	0.743; 0.786	3
	BaggedGBDT	0.876; 0.807	0.871; 0.855	0.901; 0.889	3
Imputation: MEAN	RFsklearn	0.877; 0.823	0.910; 0.836	0.731; 0.782	3
Base	RFsklearn	0.871; 0.816	0.917; 0.843	0.754; 0.760	3
Optimized	RFsklearn	0.848; 0.823	0.902; 0.819	0.849; 0.831	4
	GBDT	0.884; 0.849	0.961; 0.902	0.911; 0.859	4
Imputation: KNN5	BaggedGBDT	0.881; 0.819	0.781; 0.821	0.831; 0.831	4
Imputation: MICE	RFsklearn	0.865; 0.844	0.791; 0.822	0.774; 0.786	4
	GBDT	0.870; 0.852	0.773; 0.836	0.754; 0.790	4
Imputation: KNN5	RFsklearn	0.887; 0.823	0.824; 0.814	0.732; 0.743	4
Optimized	LR	0.800; 0.871	0.693; 0.751	0.631; 0.649	5
Ensembled MI	GBDT	0.827; 0.806	0.760; 0.799	0.780; 0.770	5
	LR	0.795; 0.839	0.729; 0.758	0.808; 0.752	5
Base	LR	0.798; 0.857	0.617; 0.689	0.601; 0.657	6
Imputation: MICE	LR	0.809; 0.788	0.602; 0.613	0.661; 0.671	6
Imputation: MEAN	LR	0.798; 0.857	0.617; 0.689	0.601; 0.657	6
SHAP zeroing	LR	0.798; 0.857	0.617; 0.689	0.601; 0.657	6
Imputation: KNN5	LR	0.807; 0.870	0.639; 0.636	0.629; 0.583	6
SHAP zeroing optimized	GBDT	0.868; 0.874	0.939; 0.812	0.889; 0.865	6

TABLE 3.2: AUCs of different base models and experiments. Each cell shows the AUC obtained by using only clinical features and when using both clinical and visual features, in that order. Each model is ranked based on one-sideded DeLong tests and the results of this ranking are shown in the 'Rank' column.

seems that the base learners all perform quite well, except for LR, though this model still performs reasonably well when predicting on both clinical and visual features. The optimized versions seem to generally perform less well than their unoptimized counterparts, except for LR. In general, the addition of visual features does not seem to improve performance and often even leads to reductions in performance. However, models that perform poorly when provided only with clinical features seem to benefit from visual features the most.

For iCTCF, we also computed the AUC in the same way as described in Ning et al., 2020 using an optimized GBDT model, which resulted in a performance of 0.942 when predicting on only clinical features (Ning et al., 2020 obtained 0.882) and an AUC of 0.965 when using both clinical and visual features (Ning et al., 2020 obtained 0.978). Note that they used a visual feature extractor trained on that same dataset, whereas ours was trained on data from Radboudumc.

3.2 Cross-hospital evaluations

Experiment	Dataset Model	CWZ: CWZ	CWZ: RUMC	iCTCF: Liyuan	iCTCF: Union	Rank
	CORADS-AI	0.885	0.826	0.812	0.854	-
Imputation: KNN5	BaggedGBDT	0.888; 0.856	0.773; 0.851	0.687; 0.862	0.688; 0.811	1
	GBDT	0.842; 0.846	0.743; 0.848	0.666; 0.842	0.613; 0.772	2
Imputation: MEAN	BaggedGBDT	0.878; 0.856	0.736; 0.836	0.724; 0.873	0.682; 0.790	2
	GBDT	0.862; 0.857	0.697; 0.845	0.713; 0.875	0.640; 0.787	3
SHAP zeroing	GBDT	0.860; 0.846	0.776; 0.860	0.644; 0.849	0.711; 0.777	4
	BaggedGBDT	0.879; 0.852	0.777; 0.836	0.637; 0.854	0.753; 0.731	4
Ensembled MI	GBDT	0.874; 0.854	0.756; 0.840	0.643; 0.863	0.667; 0.868	4
Base	BaggedGBDT	0.882; 0.854	0.736; 0.823	0.675; 0.798	0.759; 0.713	4
	GBDT	0.858; 0.840	0.725; 0.807	0.761; 0.864	0.717; 0.730	4
Imputation: MICE	GBDT	0.873; 0.855	0.694; 0.826	0.519; 0.879	0.624; 0.726	4
Optimized	LR	0.796; 0.890	0.779; 0.889	0.621; 0.861	0.502; 0.777	5
Ensembled MI	LR	0.850; 0.840	0.705; 0.835	0.657; 0.834	0.607; 0.760	5
Imputation: MICE	LR	0.774; 0.841	0.758; 0.859	0.580; 0.846	0.488; 0.552	6
Base	LR	0.897; 0.896	0.703; 0.810	0.562; 0.800	0.489; 0.596	7
Optimized	GBDT	0.886; 0.838	0.751; 0.808	0.577; 0.848	0.680; 0.734	7
Imputation: MEAN	LR	0.897; 0.896	0.703; 0.810	0.562; 0.800	0.489; 0.596	7
Imputation: KNN5	LR	0.854; 0.741	0.696; 0.836	0.634; 0.811	0.517; 0.513	7
SHAP zeroing	LR	0.897; 0.896	0.703; 0.810	0.562; 0.800	0.534; 0.567	7
SHAP zeroing optimized	GBDT	0.893; 0.863	0.809; 0.867	0.525; 0.889	0.670; 0.760	7

TABLE 3.3: AUCs of different base models and experiments when being trained on one hospital and evaluated on the other. The hospitals listed indicate which hospital the models were evaluated on. Each cell shows the AUC obtained by using only clinical features and when using both clinical and visual features, in that order. Each model is ranked based on one-sideded DeLong tests and the results of this ranking are shown in the 'Rank' column.

Table 3.3 shows evaluations of all baseline models trained on one hospital of either the CWZ or iCTCF dataset and evaluated on the other hospital within that same dataset. It shows both the performance of the models with mostly default hyperparameters, as well as with hyperparameters obtained by means of hyperparameter optimization with TPE. These experiments are listed as 'Base' and 'Optimized'. Just as in table 3.2, all models in this table were ranked using one-sided DeLong tests. Performance for the iCTCF dataset seems to be significantly worse than the previous evaluations (which did not use train/test sets based on hospital origin), although adding visual features seems to increase performance. Performance of models with optimized hyperparameters also often seems to be reduced compared to their base counterparts.

3.3 Evaluating the effects of missing data

Table 3.4 shows the AUCs of the base models when trained on missingness indicators (true when missing, false when present), without being provided with any of the numerical values in the original datasets. As can be seen in this table, the models

Dataset Model	CWZ	iCTCF	RUMC
BaggedGBDT	0.667; 0.666	0.954; 0.955	0.879; 0.885
GBDT	0.702; 0.702	0.932; 0.942	0.863; 0.863
LR	0.634; 0.633	0.943; 0.944	0.853; 0.853

TABLE 3.4: AUCs for different models when predicting PCR outcome/diagnosis based on missingness indicators only. Each cell shows the AUC obtained by using only clinical features and when using both clinical and visual features, in that order.

still perform the same or even better than when used on the measured numerical and categorical values for both the iCTCF dataset and the CWZ dataset.

Figures 3.1, 3.2, and 3.3 show the performance of each of the baseline models in predicting missingness of all features in each dataset given some imputed version of the data. These figures suggest that GBDT is capable of extracting missingness for many of the features in each of these datasets, though certain imputers seem to result in greater reductions of performance. Specifically, the k-nearest neighbour imputers, as well as to a certain extent the MICE imputer, seem to result in reduced performance (which is desirable in this case) in both the iCTCF dataset and the CWZ dataset, although none of the imputers seems to result in particularly great reductions in performance in the RUMC dataset.

The last evaluation of the extent to which models fit on missingness is that of the average absolute SHAP values. Figures B.1, B.2, and B.3 show the average absolute SHAP values of each of the baseline models for all missing features, present features, and visual features (where applicable). These figures show that across all datasets and all baseline models, there is some dependence on missing values. However, the effect is most pronounced in the iCTCF dataset.

3.4 Preventing fitting on missingness

This final section shows the results of the various methods we implemented to try to reduce the amount of fitting on missingness. They are firstly evaluated on a toy dataset (specifically the breast cancer dataset distributed with scikit-learn), which was repeatedly augmented in order to simulate problematic missingness patterns.

Figure B.4 shows the performance of a baseline GBDT model, mean imputation, 5-nearest neighbours imputation (KNN5), MICE imputation (MICE), a custom decision tree implementation (DT), SHAP-zeroing using a GBDT model, an ensemble multiple imputation model (MI), and an ensembled multiple imputation model using test-time multiple imputations (MI TT). This figure shows that the baseline GBDT model, mean imputation, and 5-nearest neighbours generally generate (much) worse-than-random predictions, whereas the custom DT, MI ensemble, and test-time MI ensemble models generally show better-than-random predictions. SHAP-zeroing and MICE imputation performance seems to vary quite a lot, though it is generally worse than random.

Based on the accuracy distributions for each model, we ran paired one-sided t-tests with Bonferonni correction. The results of this can be seen in table B.18. Table B.19 shows a ranking of the models based on the p-values ($p < 0.01$ is considered significant).

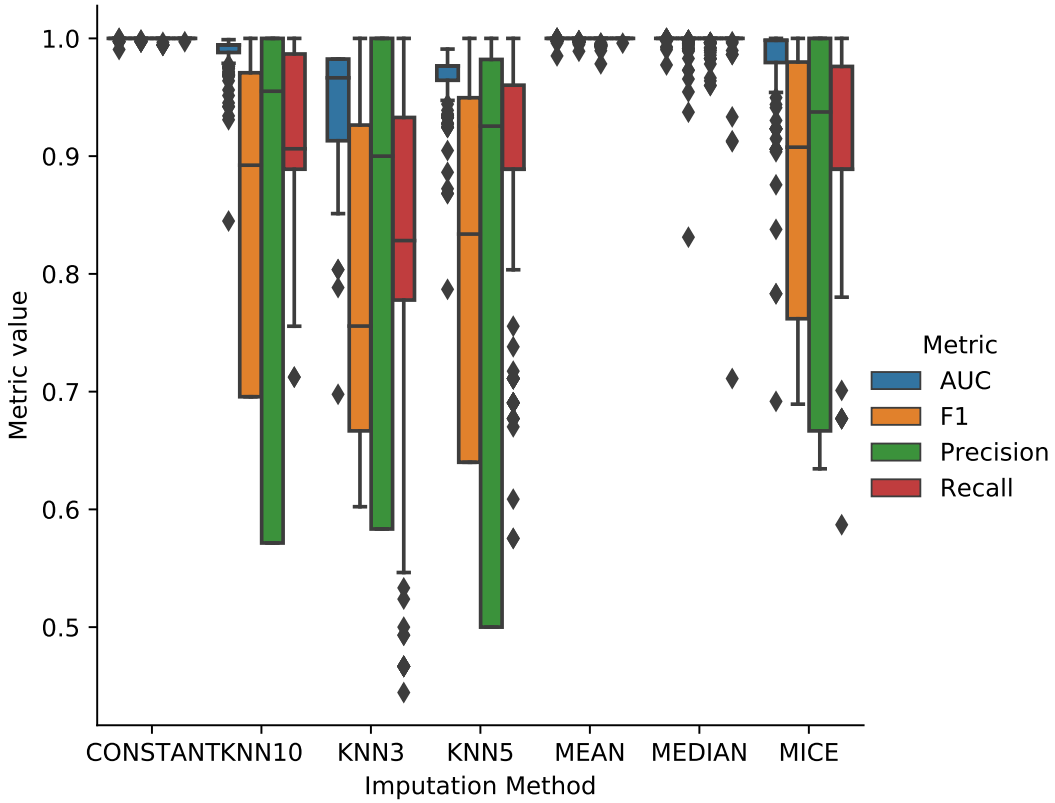


FIGURE 3.1: Missingness classification performance on the iCTCF dataset of baseline models. Models were trained on data from the first cohort and the evaluations shown here were computed based on data from the second cohort.

TABLE 3.5: P-values of one-sided paired t-test between the accuracies of all models evaluated on 100 random augmented datasets.

Second First	DT	GBDT	KNN5	MI	MI TT	MICE	Mean	SHAP zeroing
DT	-	0.000	0.000	1.000	1.000	0.000	0.000	0.000
GBDT	1.000	-	1.000	1.000	1.000	1.000	0.943	1.000
KNN5	1.000	0.000	-	1.000	1.000	1.000	0.000	0.818
MI	0.000	0.000	0.000	-	0.154	0.000	0.000	0.000
MI TT	0.000	0.000	0.000	0.846	-	0.000	0.000	0.000
MICE	1.000	0.000	0.000	1.000	1.000	-	0.000	0.000
Mean	1.000	0.057	1.000	1.000	1.000	1.000	-	1.000
SHAP zeroing	1.000	0.000	0.182	1.000	1.000	1.000	0.000	-

We then applied these techniques to the real-world COVID-19 data. Specifically, we tested them in cross-hospital prediction for both the iCTCF and CWZ datasets, as this is where we hoped to improve performance. The results of this are all shown in table 3.3, in addition to the previously mentioned 'Base' and 'Optimized' experiments. These experiments were also performed on the other train/test splits, as

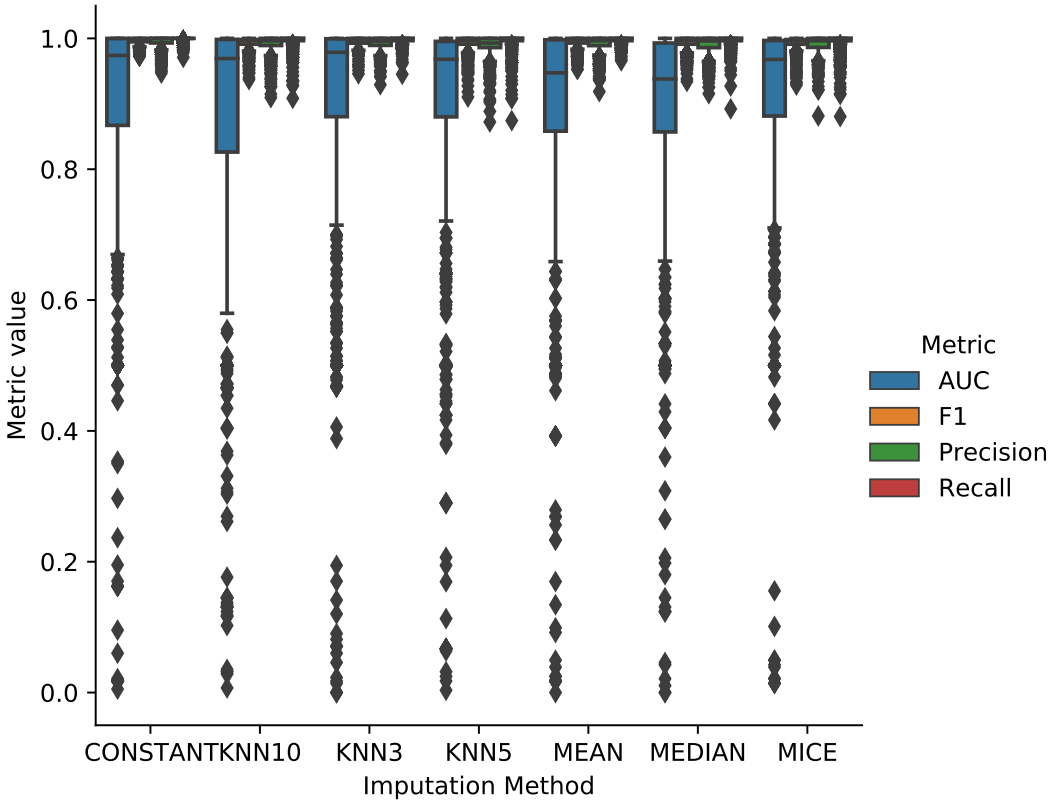


FIGURE 3.2: Missingness classification performance on the RUMC dataset of baseline models. Models were trained on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

TABLE 3.6: Ranking of models in synthetic data.

Rank	Model
1	MI
1	MI TT
2	DT
3	MICE
4	SHAP zeroing
4	KNN5
5	GBDT
5	Mean

shown in table 3.2. Firstly, we ran imputation experiments. These are shown in table 3.3 as the experiments 'Imputation: KNN5' for 5-nearest neighbours imputation, 'Imputation: MICE' for MICE imputation, and 'Imputation: MEAN' for mean imputation. As you can see in this table, all imputation techniques perform quite well, both in cross-hospital prediction as well as the default train/test splits. KNN imputation seems to perform the best in cross-hospital prediction, and mean imputation achieves the best performance for the default train/test splits, though it does not

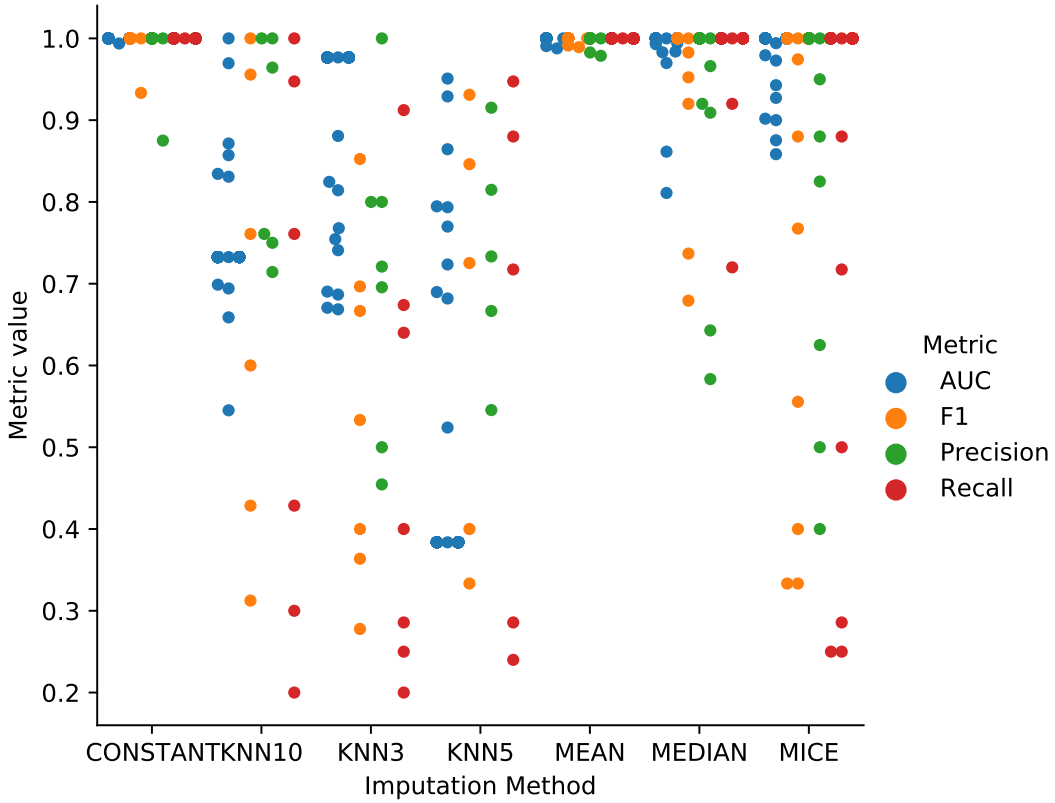
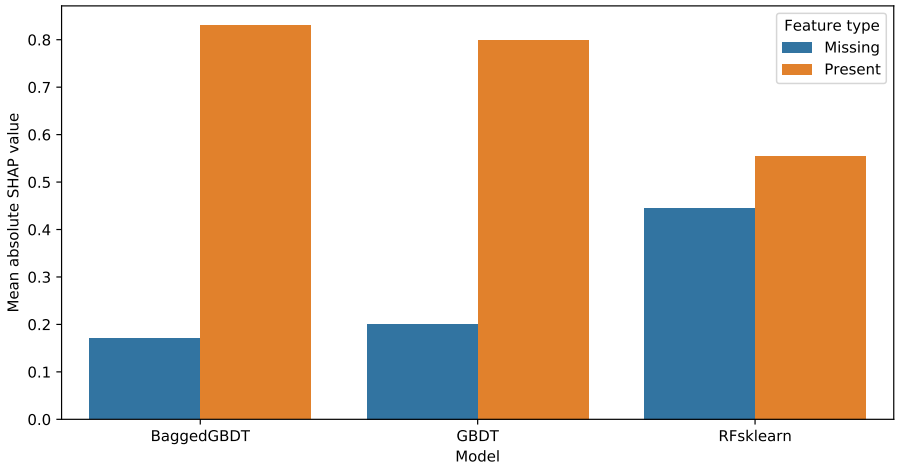


FIGURE 3.3: Missingness classification performance on the CWZ dataset of baseline models. Models were trained on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

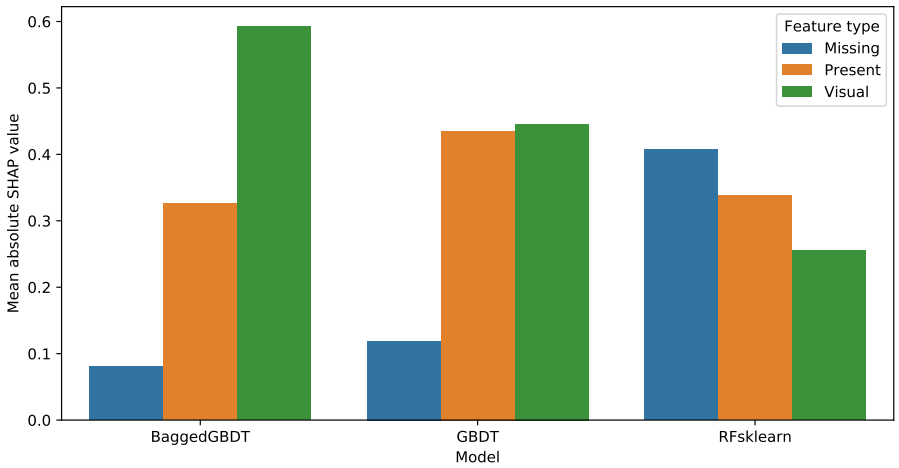
seem to outperform the base GBDT and BaggedGBDT models.

Next, we tested both SHAP zeroing, as well as SHAP zeroing with hyperparameter optimization. These are shown in table 3.3 as 'SHAP zeroing' and 'SHAP zeroing optimized', respectively. Since SHAP zeroing only changes model predictions, but does not affect training, one might expect the performance to be worse when the patterns of missingness are similar between the training and test splits. This is because the model is unable to take into account missing values, even if the model had learned to do so during training. Indeed, in the default train/test splits, SHAP zeroing leads to reduced performance, though the effect does not appear to be particularly large. In cross-hospital prediction, SHAP zeroing seems to perform similarly to the base models. SHAP zeroing with hyperparameter optimization generally seems to perform poorly.

Finally, we tested ensembled multiple imputation. These are shown in table 3.3 as 'Ensembled MI'. Similarly to SHAP zeroing, this is intended to prevent models from fitting on missingness. Once again, ensembled multiple imputation results in poorer performance in the default train/test splits. In cross-hospital prediction, the technique appears to perform similarly to the base models, though the LR model seems to improve in performance slightly. We also tested this approach using only the first cohort of the iCTCF dataset, as done by Ning et al., 2020, and obtained an AUC of 0.861 when using only clinical features and an AUC of 0.961 when using both clinical and visual features.

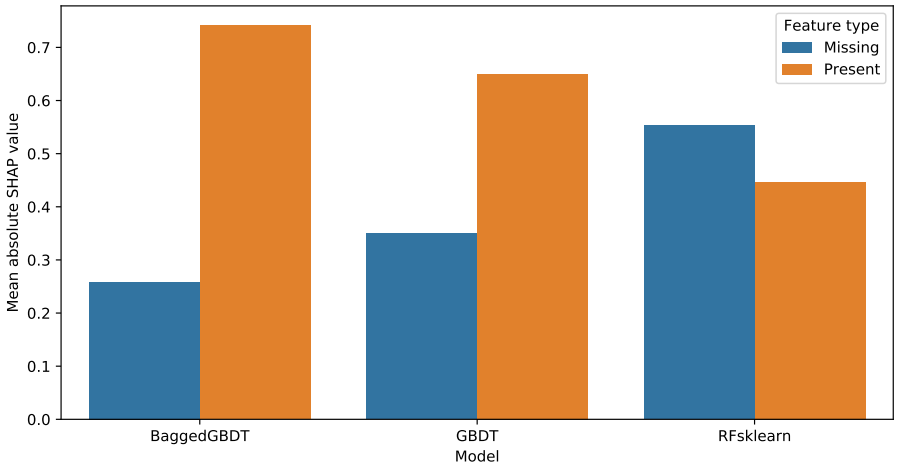


(A) Model performance based on clinical features alone.

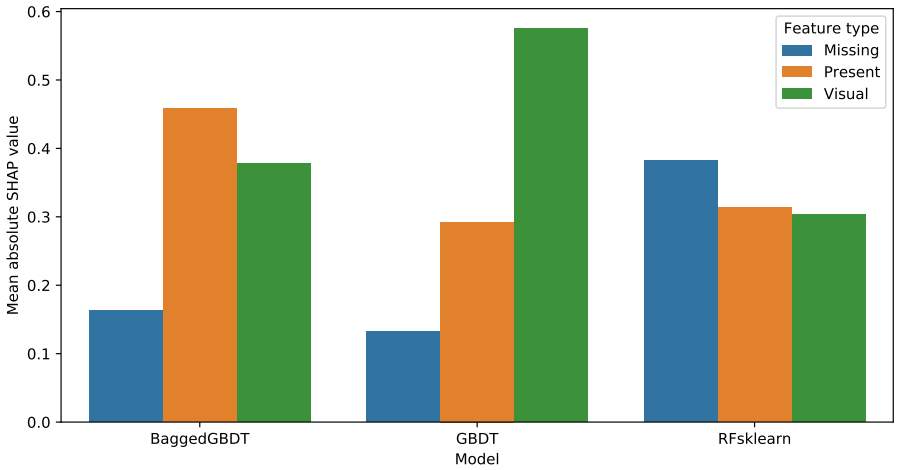


(B) Model performance based on both clinical features and visual (CT) features.

FIGURE 3.4: Mean absolute Shapley values when trained on the iCTCF dataset of baseline models. Models were trained on data from the first cohort and the evaluations shown here were computed based on data from the second cohort.

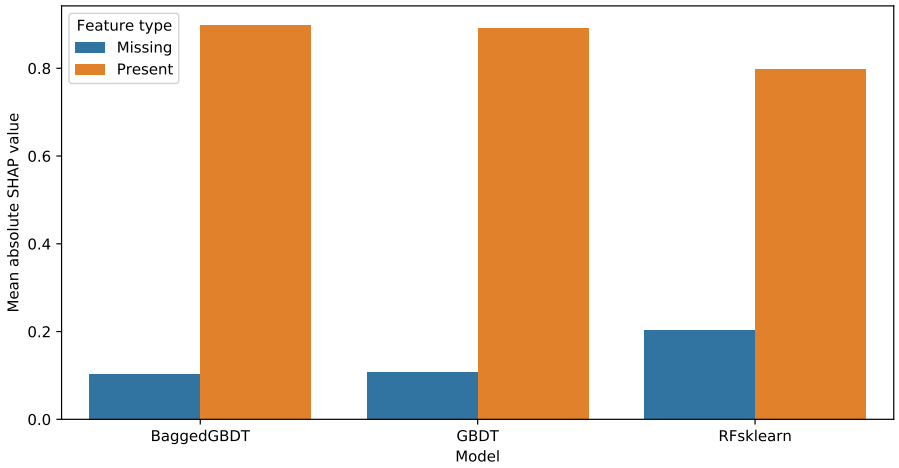


(A) Model performance based on clinical features alone.

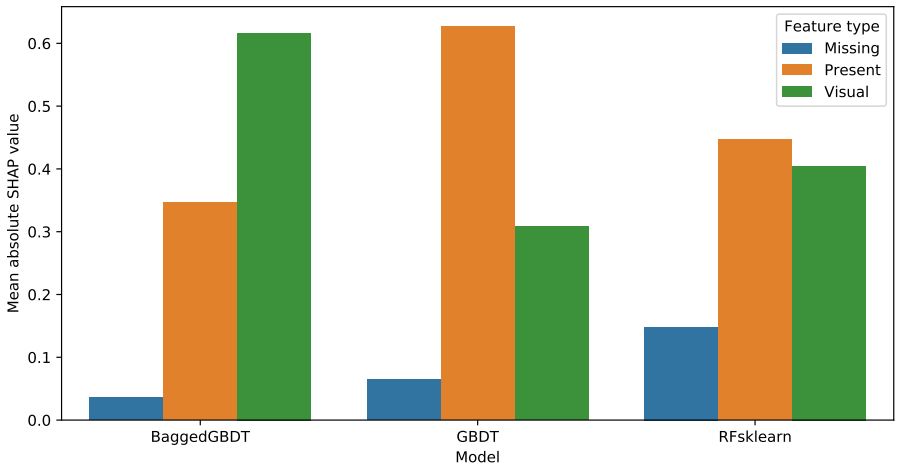


(B) Model performance based on both clinical features and visual (CT) features.

FIGURE 3.5: Mean absolute Shapley values when trained on the RUMC dataset of baseline models. Models were trained on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.



(A) Model performance based on clinical features alone.



(B) Model performance based on both clinical features and visual (CT) features.

FIGURE 3.6: Mean absolute Shapley values when trained on the CWZ dataset of baseline models. Models were trained on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

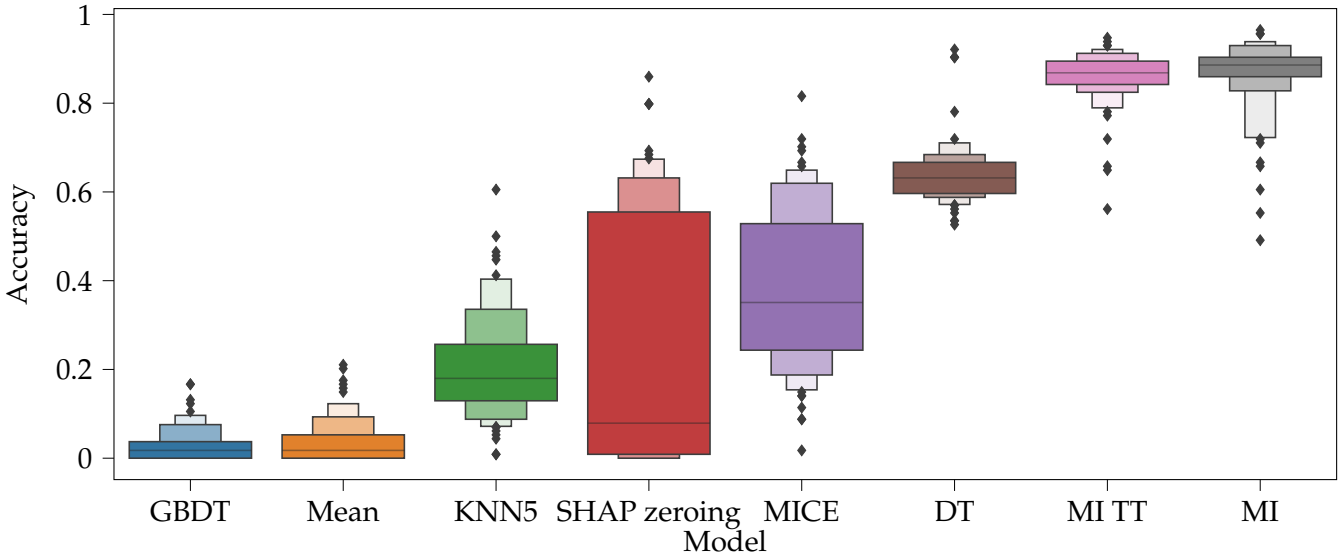


FIGURE 3.7: Accuracy scores of 100 randomly augmented versions of the breast cancer dataset for various models.

Chapter 4

Discussion and conclusions

4.1 Predicting COVID-19 diagnosis

The results show that even using the default parameters provided by the scikit-learn and LightGBM packages, most models are able to classify COVID-19 with high accuracy in terms of AUC in the iCTCF and CWZ dataset, as well as reasonable performance on the RUMC dataset. However, cross-hospital prediction for both the iCTCF and CWZ datasets led to decreased performance. This effect was most severe in the iCTCF dataset, whereas the cross-hospital prediction in the CWZ dataset still resulted in decent performance. Nonetheless, the results suggest that clinical features can indeed be used to predict COVID-19 PCR outcomes and, especially when using larger, multi-hospital datasets, could be sufficiently performant for clinical usage. Additionally, the usage of gradient boosting models seems to result in better performance than neural networks when predicting based only on clinical features, as our models were able to outperform the results reported in Ning et al., 2020. These results answer the first research subquestion (How accurately can COVID-19 be predicted based on clinical features alone?).

By combining clinical features with visual features, we were able to obtain more robust performance. While performance for the default train/test splits was often not improved by the addition of visual features, performance in cross-hospital was sometimes markedly improved. In general, performance seems to increase when CORADS-AI performs similarly well on the data or better than the performance obtained by predicting based on clinical features alone. For example, CORADS-AI achieves an AUC of 0.888 on the first cohort of iCTCF (when imputing missing visual features for missing CT scans) and when evaluating in the same way as Ning et al., 2020, we obtained similar performance with the visual features of CORADS-AI (an AUC of 0.965), improving over the model using only clinical features. This suggests that the combination of visual and clinical features is indeed possible and helpful, but perhaps unsurprisingly depends on the quality of both feature sets. Thus, it would seem that the addition of visual features is generally useful, improving performance when both the clinical and visual performance is good and preventing significant decreases in performance when either modality fails. These results answer the second research subquestion (How can clinical and visual features be combined to improve COVID-19 prediction?) as well as subquestion three (Can standard machine learning models trained on COVID-19 data from one hospital generalize to data from a different hospital?).

All in all, most evaluations resulted in high performance in terms of AUC, which suggests that these methods may indeed be clinically useful. Some further considerations regarding clinical usage are discussed in section 4.7.

We suspected one reason for the much-decreased performance in cross-hospital

prediction in iCTCF, as well as potentially calling into question some of the performance results obtained, was the missingness present in the datasets' clinical features. More specifically, we suspected the missingness of clinical features was in and of itself indicative of the patient's diagnosis. We therefore designed and performed several experiments to answer this question.

4.2 Determining the predictiveness of missingness

We performed three experiments to determine the extent to which missingness was used by the various baseline models in predicting the target label. Firstly, we designed an experiment where all numerical values in each of the datasets were replaced with missingness indicators and trained models to predict the target labels based on these missingness labels alone. Especially on the iCTCF dataset, this resulted in performance comparable to that of the models trained directly on the numerical data present in this dataset. This shows that whether values were measured is no less predictive than the measured values themselves, answering the fourth research subquestion (are the patterns of missing data in COVID-19 clinical data predictive of the diagnosis?).

While these results indicate that missingness is indeed predictive of the target label, it does not prove that the previous models trained on the original datasets were depending on the information. It is possible that the measured values themselves are just as predictive of the target label as the missingness of the values. However, it is also possible that the missingness is the only feature particularly predictive of the target label. All of the models provided by the LightGBM package receive unimputed data. That is, the data they are given directly indicate whether certain values are missing. Therefore, it is possible that the models depend, to some extent, on these patterns of missingness. Even when missingness is removed by means of imputation, as is the case of all scikit-learn models, it might still be possible to deduce which values were missing or not. Our second experiment sought to determine the extent missingness can be obscured by various imputation techniques.

To this end, we trained multi-output models to predict missingness given imputed data. Results suggest that, while some imputation techniques like k-nearest neighbours seem to perform better than others, there are generally many features of which the missingness is trivially predicted.

These results suggest that these imputation techniques fail to properly mask the missingness in the data. However, there are other interpretations possible. Since we do not expect values to be missing completely at random, as clinicians were involved in determining what values to measure, there must be some structure to which values are missing or not. If one had some imaginary perfect imputation technique that was capable of filling in missing values as though they had been measured in the first place, on might still expect the performance of the models in predicting missingness to be above random. This is because the model would be able to perform the same task as the clinician in selecting which values to measure or not, especially since it would have an information advantage to the clinician (it already knows all values, whereas the clinician does not). Thus, this experiment fails to provide conclusive evidence, though some techniques, such as mean imputation, do seem to be much inferior at masking missingness.

We conducted one final experiment to determine the extent to which models fit on missingness. This time, we did not train any new models. Rather, we simply analyzed the ones we had already trained. Using the shap Python package, we

computed SHAP values for each of the trained models and aggregated all values associated with missing, present, and visual features by taking an average of the absolute values of each of these groups. These results show that, especially for the iCTCF dataset, models tend to depend at least partially on missing values for their predictions. In the ideal case, we had expected the SHAP values associated with missing values to be zero, as they should not impact model predictions. However, this appears not to be the case. All these results together suggest that fitting on missingness is likely present, which in and of itself is undesirable, and may also explain—at least partially—why the cross-hospital performance is so significantly decreased in the iCTCF dataset, as such missingness patterns may differ between hospitals.

4.3 Preventing fitting on missing data

To remedy the issue of fitting on missing data, we developed and adopted several strategies for dealing with missing data: imputation, SHAP zeroing, multiple imputations, and native missingness handling in decision trees.

4.3.1 Synthetic data

In order to determine how well these different techniques work, we evaluated them on a synthetic test dataset. The results suggest that multiple imputations and native missingness handling in decision trees work very well to prevent fitting on missingness, while SHAP zeroing helps moderately. Simple single imputations do not appear to improve performance here or even result in reduced performance. This answers the sixth research subquestion (How can overfitting on patterns of missing data be mitigated?).

4.3.2 Real data

While this synthetic example provides some evidence of the performance of the techniques, we are ultimately interested in whether it actually improves performance in cross-hospital prediction. To this end, we ran cross-hospital prediction evaluations with each of these techniques. The results from these evaluations suggest that none of the techniques leads to the significantly improved performance that we had hoped for. For both GBDT and BaggedGBDT, however, SHAP zeroing results in reduced performance on the default train/test splits. As this technique only removes those parts of a model's prediction based on missing values, this suggests that these models do indeed fit slightly on missingness, though SHAP zeroing suggests the effect isn't that large. However, ensembled multiple imputation does lead to much-reduced performance. As this technique is also intended to reduce fitting on missingness—and its performance on the augmented datasets seems to suggest it indeed manages to achieve this–this suggests that the component of missingness is indeed quite important. This, in turn, suggests that SHAP zeroing fails to properly mask missingness. Regardless, it seems that these methods only perform well on synthetic data, but fail to improve performance on the real-world problem we are trying to solve.

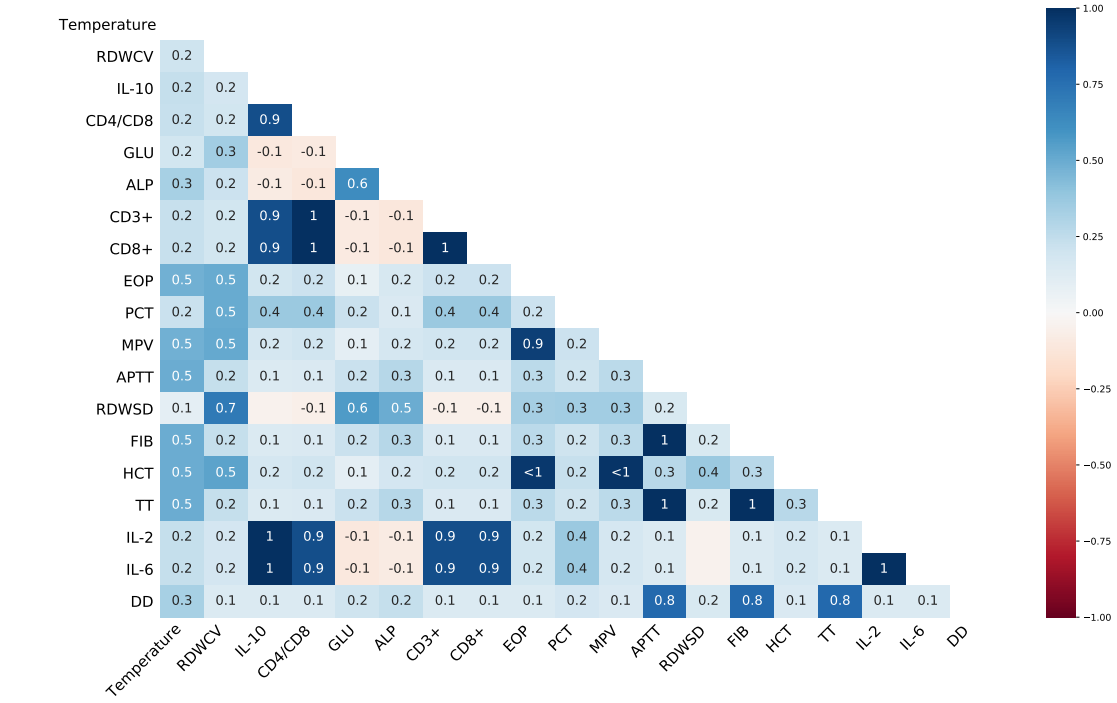


FIGURE 4.1: Missingness correlations between of top 20 features of iCTCF.

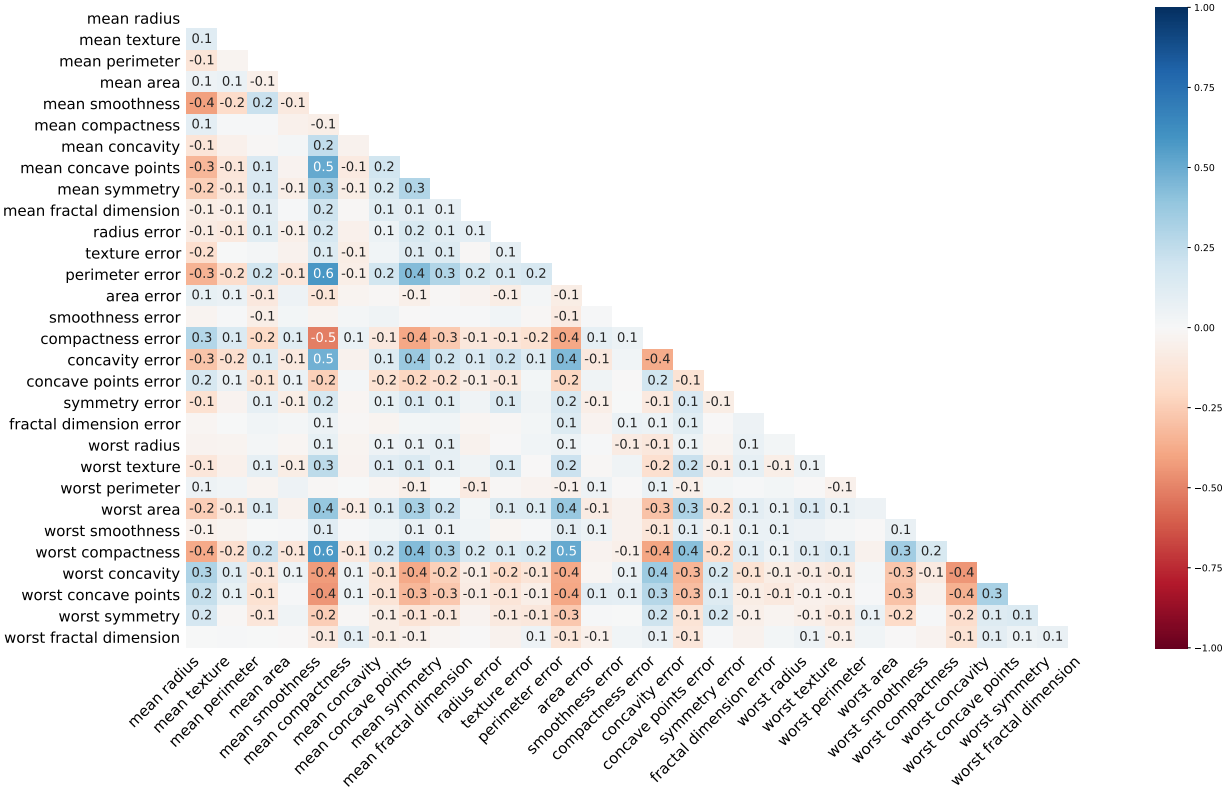


FIGURE 4.2: Missingness correlations between features randomly augmented (as described in section 2.6.1) breast cancer data.

TABLE 4.1: Top-15 features based on mean absolute SHAP values of the Union and Liyuan hospitals in the iCTCF dataset.

	Union	Liyuan
1	Coefficient variation of red cell volume distr...	Metadata_Temperature
2	Metadata_Temperature	Metadata_Age
3	Interleukin-10 Value	Calcium Value
4	CD4/CD8 ratio Value	Mean platelet volume Value
5	CD3+ T cell Value	Activated partial thromboplastin time Value
6	Standard deviation of red cell volume distribu...	Alkaline phosphatase Value
7	CD8+ T cell Value	D-Dimer Value
8	Glucose Value	Chlorine Value
9	C-reactive protein Value	High-sensitivity C-reactive protein Value
10	Platelet distribution width Value	Uric acid Value
11	IFN-y Value	Eosinophil percent Value
12	Eosinophil percent Value	Antithrombin III Value
13	Monocyte percent Value	Total carbon dioxide Value
14	Metadata_Age	Glucose Value
15	Plateletcrit Value	Neutrophil count Value

4.4 Alternative causes of reduced performance in cross-hospital prediction

There are various possible interpretations as to why the performance improvements fail to generalize to real-world datasets. Firstly, it is possible that the synthetic nature of the dataset fails to capture the exact issues present in the real-world data. The way we generated the synthetic missing data was by generating random missingness probability vectors for either of the two possible classes in this dataset and swapping these vectors between the train and test datasets. However, the missingness probabilities may not only be dependent on the output label, but also on other features in the dataset and features that are not recorded in this dataset at all. For example, a doctor may have observed some signs and symptoms in a patient, but only blood values were present in the iCTCF dataset; thus the doctor's decision to measure a certain value might be dependent on these features, without these features being noted in the dataset. We can see this property of missingness correlation between features in the iCTCF dataset in the heatmap in figure 4.1. Additionally, figure 4.2 shows a similar heatmap, but for an augmented version of the breast cancer dataset.

Another potential issue is that the missingness probability vectors are generated uniformly at random, and this process is repeated many times. This might result in many probability distributions that are more favourable than real-world distributions. For example, in an extreme case, if in real-world data we (almost) only measure the first $n/2$ features for the first hospital and (almost) only the second $n/2$ features for the second hospital, then it would be difficult or even impossible to use the data from one hospital to predict the labels of the second. In this scenario, fitting on missingness is not so much the problem. Rather, the data is simply of insufficient quality to perform the task.

Of course, in real-world data, things are not so black and white as in this previous example scenario. However, even when the problem is not so pronounced, it may still be present more subtly. That is, there may be highly predictive features that

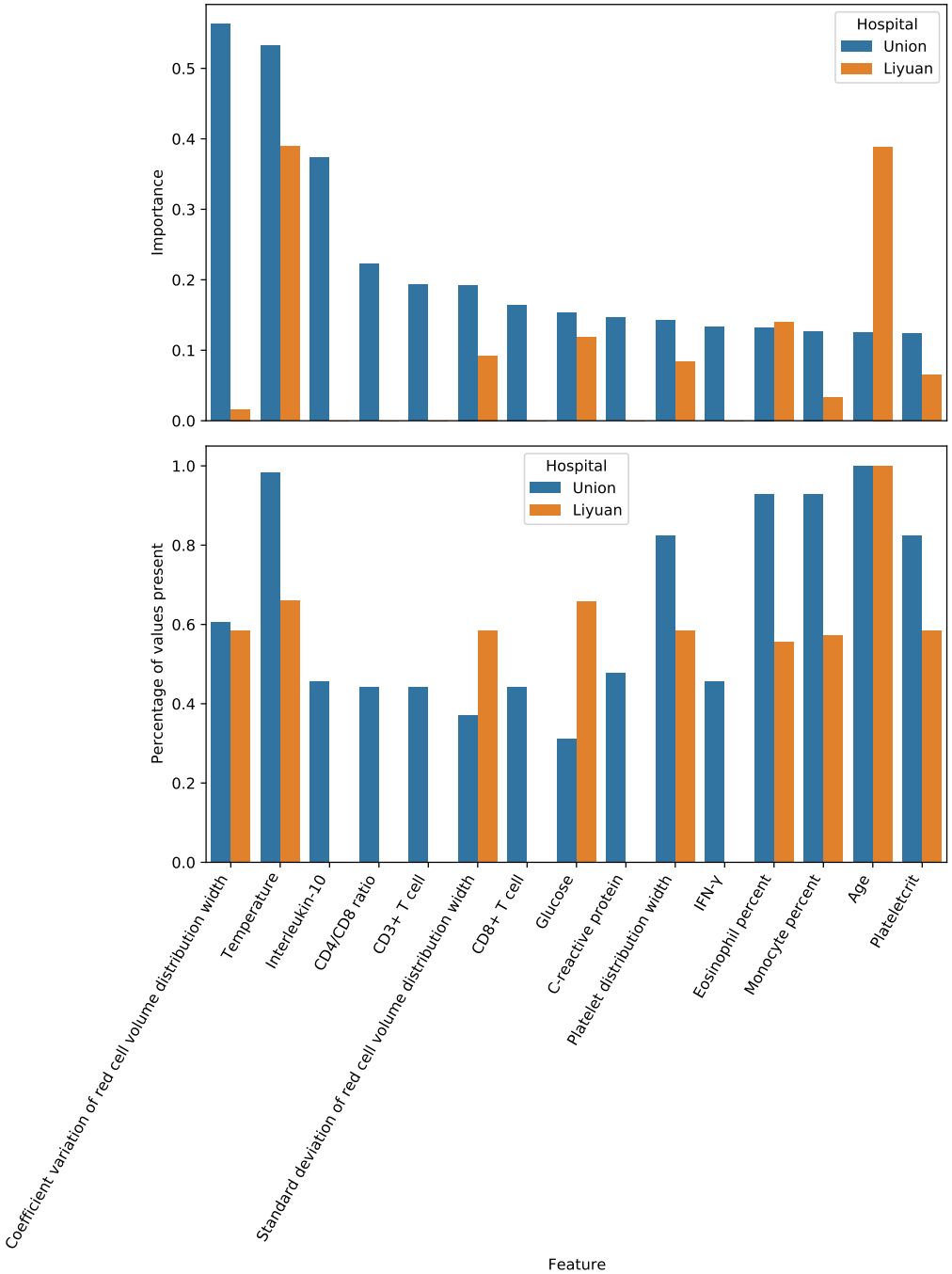


FIGURE 4.3: Top-15 features in the Union hospital data in the iCTCF dataset based on mean absolute SHAP values. For each feature, the mean absolute SHAP values of all present values are shown for both Union and Liyuan, as well as the percentage of feature values present for both hospitals.

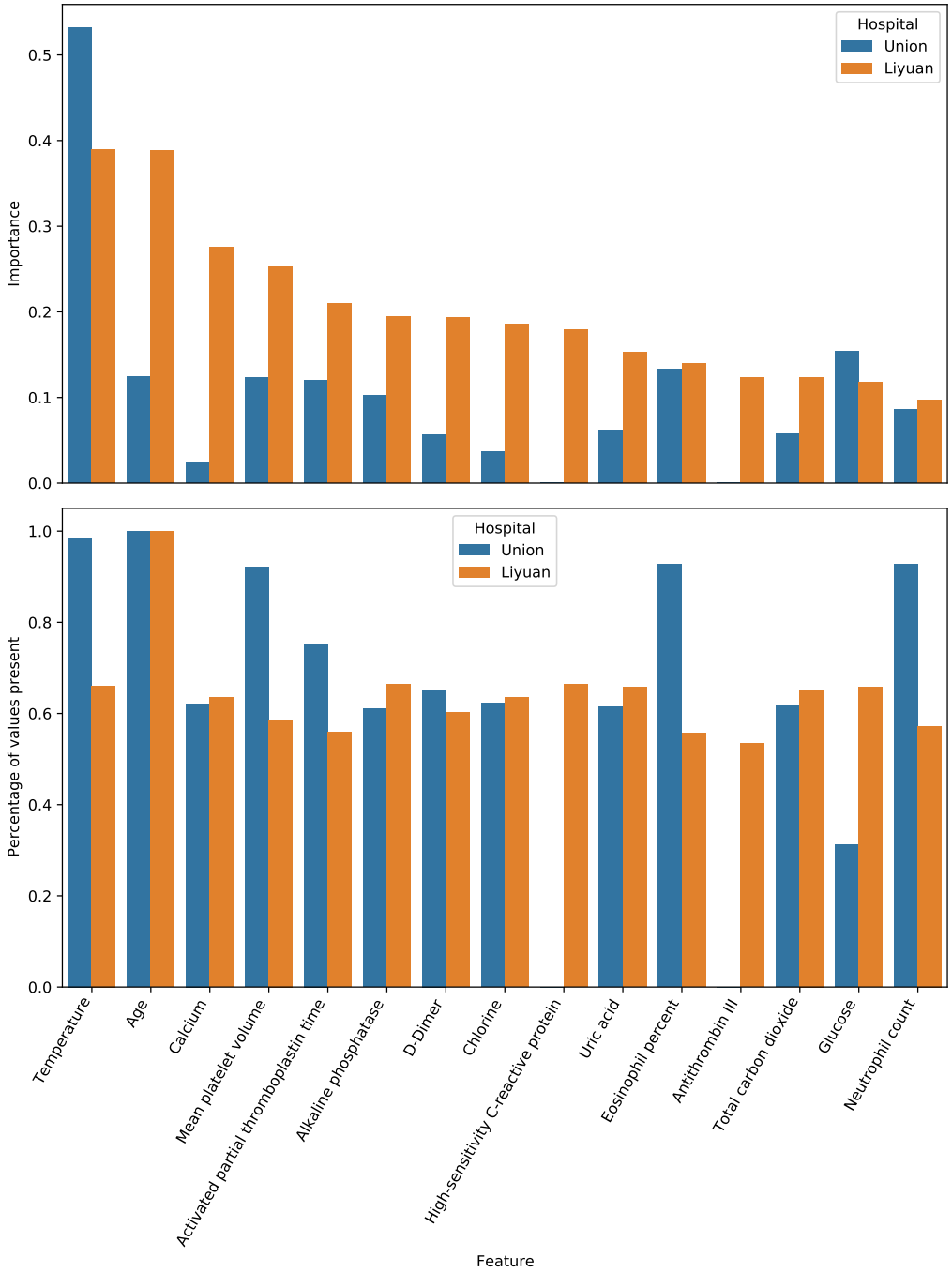


FIGURE 4.4: Top-15 features in the Liyuan hospital data in the iCTCF dataset based on mean absolute SHAP values. For each feature, the mean absolute SHAP values of all present values are shown for both Union and Liyuan, as well as the percentage of feature values present for both hospitals.

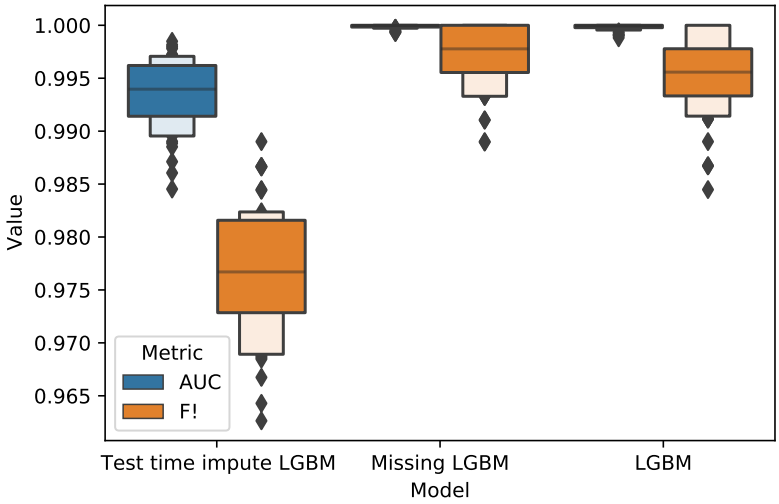


FIGURE 4.5: AUC and F1-score prediction performance of three classifiers trained to predict the hospital of origin of clinical feature data in the iCTCF dataset.

are commonly measured in one hospital and highly predictive features commonly measured in the other, but if these feature sets do not overlap strongly, we may run into the same issue. To evaluate this potential issue, we determined the twelve most important features in the Union hospital and the Liyuan hospital data in the iCTCF dataset by means of mean absolute SHAP values per feature. These are shown in table 4.1. Additionally, figures 4.3 and 4.4 show the mean absolute SHAP values as well as the missingness for both hospitals for the top 15 features of the Union and Liyuan hospitals, respectively. As you can see in these figures, some features that are deemed important in the PCR classification in one hospital are simply not present in the other, meaning everything the model learned pertaining to these features is useless when classifying data from the other hospital. In addition, there are some features that are similarly present in both hospitals, yet have wildly different mean absolute SHAP values. This suggests that there may be some fundamental differences between the distribution of patients admitted to either hospital, which would explain why most models struggle to generalize between these datasets.

For example, the feature deemed most important in the Union data is the coefficient variation of the red blood cell distribution width (RDW-CV). Lippi, Henry, and Sanchis-Gomar, 2021 show that this feature is particularly indicative of severe illness in COVID-19. Since the iCTCF dataset provides morbidity data, we can look at the distribution of morbidity between the two hospitals, see figure 2.3. This figure shows that the Union hospital received more severely ill patients, potentially explaining this difference in predictiveness of RDW-CV.

More generally, to determine to what extent the distributions of patients in the two hospitals differ, we trained three classifiers to predict the hospital that the input clinical feature data originated from. The classifiers we used were a simple GBDT model (which would base itself on both missingness and feature values), a GBDT model using missingness indicators (which would base itself only on missingness), and an ensembled multiple imputation GBDT model with test-time multiple imputation (which should base itself only on feature values). We ran this test with 10-times repeated 5-fold cross-validation. The results of this test are shown in figure

4.5. As you can see in this figure, it appears that both the missingness as well as likely feature values themselves are highly predictive of the hospital of origin. This gives further evidence to the idea that there is a consistent difference in the distribution of the two datasets.

Finally, another explanation might be that certain features are highly correlated, resulting in a model fitted on one hospital using one feature and the model fitted on the other hospital to fit on a highly correlated feature, but not the same feature. This would then result in a discrepancy between the mean absolute SHAP values of both models for that feature, even if the feature in question is highly predictive.

Regardless of the exact issues present in cross-hospital prediction, the fact that none of the methods we implemented and utilized managed to significantly increase performance in cross-hospital prediction suggests that, while the models likely do fit to some extent on patterns of missingness, they are not the main cause of reduced performance in cross-hospital prediction.

4.5 Implications of fitting on missing data

The last research subquestion that remains unanswered is the fifth research subquestion (do the patterns of missing data in COVID-19 clinical data harm model generalizability?). While we have established without much doubt that, especially in the iCTCF dataset, the presence or absence of a value is highly predictive of the PCR outcome and have shown that various methods can be utilized to mitigate the fitting on missingness in synthetic data, we have not clearly established whether these patterns of missingness reduce generalizability. The various experiments we have run do suggest that there is some fitting on missingness in the data. However, as discussed in the previous section, this is likely not the main cause of the reduced performance in cross-hospital prediction. However, even if this was not the main cause, we still believe that reducing fitting on patterns of missingness is important, as reporting performance without correcting for this issue will inevitably result in over-optimistic performance estimates. We wish to make predictions on clinical values themselves, not whether it was decided to measure a particular value. Missingness leaks a clinician's judgment into the dataset, giving the classifier an unfair advantage and reducing performance when such patterns are removed, such as in standardized testing procedures or potentially when different clinicians with different procedures are involved. These issues are potentially worsened even more if the values were measured after a PCR result had been obtained, as the clinician would now be basing his decision to measure it directly on the PCR outcome.

Thus, even if it is not the main cause of reduced performance in this thesis's experiments, fitting on missingness can not be ignored and inevitably harms model generalizability in this domain, answering the fifth research subquestion.

4.6 Data quality

There are many issues that could or are certainly present in the data utilized in this thesis. Some of these are the result of the preprocessing performed. Specifically, for the RUMC dataset, we performed much preprocessing as the raw data for this dataset required it. Some of the decisions made in this preprocessing may have resulted in reduced performance. One essential decision that had to be made was that of what time ranges to use to extract clinical values around each CT scan date. Choosing a small time range means the values used for each patient and study are

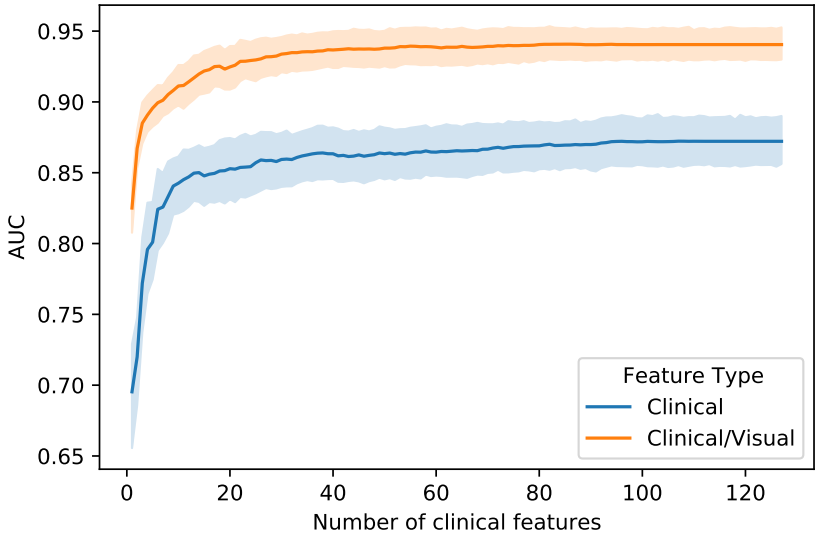


FIGURE 4.6: 10-fold cross-validated AUC for default GBDT model as a function of the number of features provided to the model when generating predictions.

more accurate and correspond more closely to the visual features from the CT scan. However, when using a small time range, the number of values available quickly becomes unworkable (i.e., thresholding based on the number of missing values results in all data being removed). Thus, in order to perform any learning, a larger range is necessary. Since these values also include PCR results, this issue becomes even more important, as we certainly need the target labels themselves in order to make a prediction.

The other two datasets, iCTCF and CWZ, also present some issues, though less preprocessing was required for them. However, as stated previously, there are some potential issues that remain about how and when values were collected, and how much external information doctors had when deciding to have certain values measured, which means certain external information may have leaked into the data. While this thesis did not look into this problem, as we only had temporal information for the RUMC dataset, it would seem likely that such information is relevant when generating ML models; thus, future studies should attempt to look into such issues more.

4.7 Clinical usage

While the models do seem to perform fairly well in most scenarios, there are more points to consider besides simple model performance. For example, some datasets, such as RUMC, have a very large number of features. Inputting and measuring such a large number of parameters may be impractical in reality. SHAP values allow us to determine which features are generally the most important. This can be used to determine which features should be given priority, but the question remains whether the performance is still sufficient when only some of the parameters are entered into the models, as there are likely many features that contribute at least a bit to the model prediction.

To study what the performance decrease resulting from only using the top-k features is, we ran a small experiment using the iCTCF dataset. The results of this are shown in figure 4.6. This figure shows the AUC of 10-fold cross-validated performance on the iCTCF dataset when using the top-k features based on the mean absolute SHAP value for each feature. It shows this for both models trained on only clinical features and on both visual and clinical features. When a model was trained with visual features as well, these features were always present in the evaluation, and the top-k clinical features were concatenated to the visual feature vector.

4.8 Future work

While much of this thesis focused on dealing with the missingness present in the data, there are many other relevant issues that were therefore overlooked in favour of this issue. In the following subsection, we will discuss some potentially important avenues of research that we were unable to pursue for this thesis.

4.8.1 Differentiable decision trees

Currently, we use a pretrained CNN to extract visual features and then concatenate these visual features with the clinical features, after which we train some ML model to make PCR outcome predictions. However, this CNN model was trained to detect CO-RADS scores rather than PCR outcomes. Additionally, the network was only ever trained to predict using chest CT scans, and there is thus no reason to extract those visual features that would only be useful in combination with clinical features. Since the network is not updated while training the second ML model, this may result in poorer performance. However, all models used in this thesis are non-differentiable (except for logistic regression), meaning it is not possible to back-propagate the error in PCR prediction to the CNN.

We had originally planned to use a differentiable relaxation of decision tree-based models to solve this problem. Specifically, one might utilize an approach similar to that described by Saberian, Delgado, and Raimond, 2019 by concatenating it to the preexisting CNN. This would then allow one to train both the visual feature extraction as well as the final classifier at the same time, which may result in better visual features and thus better final classifier performance.

4.8.2 More representative synthetic data

The synthetic data used in this thesis to evaluate the various methods for dealing with missing data were rather simple, in that there were only two missingness probability vectors for either classification label. However, as can be seen in figure 4.1, this is not entirely representative of real-world data. The real-world data shows that some features are always or often measured together. Additionally, this thesis used the commonly used breast cancer toy dataset to perform as a base, but it may be better to use a wider range of base datasets or even to use entirely synthetically generated datasets. To obtain a more realistic evaluation of the methods described in this thesis regarding their ability to prevent fitting on missingness, it would seem desirable to improve upon the synthetic data to bring it more in line with the missingness patterns present in real-world data.

4.8.3 Improving SHAP zeroing and integrating it into the loss function

SHAP zeroing seems to achieve poorer performance than both native missingness handling decision trees and ensembled multiple imputation in the augmented datasets experiment. There are two possible explanations for this discrepancy. Firstly, SHAP zeroing may not properly mitigate missingness because it only removes the contribution of missing features, but does not affect the contribution of present features at all. However, it is possible for a feature to have a zero SHAP value for all missing values and non-zero SHAP values for present values that do not vary (much) based on the underlying value. In such a case, the model would still base its predictions on missingness, even if the SHAP values are zero. One would therefore have to analyze this by looking at the variance of SHAP value with large absolute means for underlying present values, but with low variance, and determine whether one could adapt the approach to account for such things.

A second issue with the SHAP zeroing approach we employed in this thesis was that it is only involved at test-time, but not during training. There are two ways one might integrate SHAP values into the training phase. Firstly, one could add a component to the loss function based on the mean absolute SHAP values of missing feature values. This would mean that whatever optimization procedure one is using, the SHAP values associated with missing feature values would be decreased during training, although they would not necessarily be zero. Another approach would be to compute the loss function based on model predictions made using SHAP zeroing, which would ensure SHAP values associated with missing feature values would be exactly zero while still allowing the model to optimize for this change.

4.8.4 Self-supervised learning

This thesis considered only visual features extracted using CORADS-AI. However, another approach might be to use some self-supervised learning algorithm to extract generic visual features. In order to do so, however, a large–unlabeled–chest CT scan dataset would be necessary, as well as a sufficient computational budget to allow the training of such a model. The advantage, however, would be that one could obtain more generic high-level visual features, whereas the present study considered visual feature vectors that were most representative of CORADS scoring. This could then once again lead to improved classification performance, as well as serving as a good starting point for other research.

4.8.5 Predicting other features

While this thesis focused on predicting PCR outcomes and clinical diagnoses, one could also train similar models to predict other features, such as morbidity, mortality, other details about prognosis, or certain treatment outcomes. Since the iCTCF dataset provides information about mortality, we were able to look at predicting such things briefly and were indeed able to achieve high performance. We therefore believe that future investigations in such models could be fruitful, although one would, of course, need to have relevant data for such tasks. Finally, one would have to determine which features would be most useful to predict in clinical practice.

4.9 Conclusions

In this thesis, we investigated whether chest CT scans can be combined with clinical features to achieve better performance in predicting COVID-19 PCR outcomes and diagnoses than either modality on its own. Based on our experiments, this does indeed seem to be the case for most datasets and train/test splits, as long as both data modalities are of sufficient quality.

Furthermore, we discovered that which values were measured in hospitals is often very predictive of COVID-19 PCR outcome/diagnosis, suggesting that models may sometimes utilize this information rather than the numerical clinical values themselves. Going forward, such issues should be taken into account when working with clinical data like this, as this may compromise generalizability and result in skewed representations of model performance. Indeed, it would seem that there is little value in a model whose predictions are merely reflections of an expert's verdict, embedded in the data by way of missingness. Therefore, the various techniques we presented in this thesis may be beneficial in mitigating these problems and obtaining fairer representations of model performance and generalizability.

Regardless of these issues, we do believe that some of the models developed in this thesis could be clinically useful in detecting COVID-19, as well as being used for COVID-19 screening. Furthermore, the methodologies we employed could be employed for related clinical problems.

In the future, the methods we developed can be used to improve how this kind of research is conducted and evaluated, resulting in more reliable clinical machine learning models. Additionally, the methodologies we presented here may be expanded upon further in a variety of ways, which could aid future clinical and non-clinical research alike.

Appendix A

Hyperparameter Optimization Search Spaces

Parameter	Distribution	Distribution parameters
boosting	Constant	gbdt
n_estimators	Uniform integer	Between 2 and 700
learning_rate	Log-uniform	Between 1e-20 and 1.0
max_depth	Uniform integer	Between 0 and 30
num_leaves	Uniform integer	Between 2 and 280
reg_alpha	Log-uniform	Between 1e-20 and 100.0
reg_lambda	Log-uniform	Between 1e-20 and 1000.0
subsample	Uniform	Between 0.01 and 1.0
subsample_freq	Uniform integer	Between 0 and 30
min_child_samples	Uniform integer	Between 0 and 50
max_bin	Uniform integer	Between 3 and 500
extra_trees	Categorical	One of: True, False
path_smooth	Log-uniform	Between 1e-20 and 100.0
colsample_bytree	Uniform	Between 0.005 and 1.0
linear_tree	Categorical	One of: True, False
feature_fraction_bynode	Uniform	Between 0.005 and 1.0
min_gain_to_split	Log-uniform	Between 1e-20 and 100.0
min_sum_hessian_in_leaf	Log-uniform	Between 1e-10 and 1000.0

TABLE A.1: Hyperparameter search space for Gradient Boosting Decision Tree model.

Parameter	Distribution	Distribution parameters
solver	Constant	saga
penalty	Categorical	One of: l1, l2, elasticnet
tol	Log-uniform	Between 1e-10 and 0.01
C	Log-uniform	Between 1e-05 and 100.0
l1_ratio	Uniform	Between 0.0 and 1.0
max_iter	Constant	1000

TABLE A.2: Hyperparameter search space for Logistic Regression model.

Parameter	Distribution	Distribution parameters
n_estimators	Constant	200
criterion	Categorical	One of: gini, entropy
max_depth	Uniform integer	Between 1 and 42
min_samples_leaf	Uniform integer	Between 1 and 100
max_features	Categorical	One of: sqrt, log2, None

TABLE A.3: Hyperparameter search space for scikit-learn Random Forest model.

Parameter	Distribution	Distribution parameters
boosting	Constant	gbdt
n_estimators	Uniform integer	Between 2 and 50
learning_rate	Log-uniform	Between 1e-20 and 1.0
max_depth	Uniform integer	Between 0 and 30
num_leaves	Uniform integer	Between 2 and 280
reg_alpha	Log-uniform	Between 1e-20 and 100.0
reg_lambda	Log-uniform	Between 1e-20 and 1000.0
subsample	Uniform	Between 0.01 and 1.0
subsample_freq	Uniform integer	Between 0 and 30
min_child_samples	Uniform integer	Between 0 and 50
max_bin	Uniform integer	Between 3 and 500
extra_trees	Categorical	One of: True, False
path_smooth	Log-uniform	Between 1e-20 and 100.0
colsample_bytree	Uniform	Between 0.005 and 1.0
linear_tree	Categorical	One of: True, False
feature_fraction_bynode	Uniform	Between 0.005 and 1.0
min_gain_to_split	Log-uniform	Between 1e-20 and 100.0
min_sum_hessian_in_leaf	Log-uniform	Between 1e-10 and 1000.0

TABLE A.4: Hyperparameter search space for bagged Gradient Boosting Decision Tree model.

Appendix B

Full results

B.1 Within-dataset prediction

Firstly, we evaluated all base models on each of the three datasets with mostly default hyperparemters. The results of these evaluations can be seen in figures B.1, B.2, and B.3, which show the performance in terms of AUC, F1-score, precision, and recall on the iCTCF, RUMC, and CWZ datasets respectively.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.911	0.879	0.808	0.963
GBDT	**0.940**	**0.884**	**0.817**	0.963
LR	0.617	0.744	0.741	0.747
RFsklearn	0.917	0.869	0.776	**0.988**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.889	0.829	0.723	**0.971**
GBDT	**0.899**	**0.849**	**0.757**	0.967
LR	0.689	0.756	0.741	0.771
RFsklearn	0.843	0.818	0.727	0.935

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.1: Classification performance on the iCTCF dataset of baseline models. Models were trained on data from the first cohort and the evaluations shown here were computed based on data from the second cohort.

We also ran these same evaluations with hyperparemter optimized versions of the baseline models, these evaluations are shown in figures B.4, B.5, and B.6. For iCTCF, we also computed the AUC in the same way as described in Ning et al., 2020 using an optimized GBDT model, which resulted in a performance of 0.942 when predicting on only clinical features (Ning et al., 2020 obtained 0.882) and an AUC of 0.965 when using both clinical and visual features (Ning et al., 2020 obtained 0.978).

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	**0.881**	**0.667**	**0.625**	0.714
GBDT	0.859	0.667	0.545	**0.857**
LR	0.601	0.417	0.500	0.357
RFsklearn	0.754	0.105	0.200	0.071

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	**0.875**	**0.710**	0.647	**0.786**
GBDT	0.835	0.333	**0.750**	0.214
LR	0.657	0.364	0.500	0.286
RFsklearn	0.760	nan	nan	nan

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.2: Classification performance on the RUMC dataset of baseline models. Models were trained on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.869	0.857	0.814	0.905
GBDT	0.870	0.859	**0.846**	0.873
LR	0.798	0.846	0.821	0.873
RFsklearn	**0.871**	**0.866**	0.817	**0.921**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.806	0.829	0.850	**0.810**
GBDT	0.836	0.820	0.847	0.794
LR	**0.857**	**0.840**	**0.893**	0.794
RFsklearn	0.816	0.810	0.845	0.778

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.3: Classification performance on the CWZ dataset of baseline models. Models were trained on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.961**	0.822	0.698	**1.000**
LR	0.693	0.812	0.730	0.914
RFsklearn	0.902	**0.845**	**0.738**	0.988

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.902**	**0.843**	**0.756**	**0.951**
LR	0.751	0.801	0.724	0.898
RFsklearn	0.819	0.805	0.712	0.927

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.4: Classification performance on the iCTCF dataset of baseline models. Models were trained and hyperparameters were optimized on data from the first cohort and the evaluations shown here were computed based on data from the second cohort.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.911**	0.686	0.571	**0.857**
LR	0.631	0.400	0.455	0.357
RFsklearn	0.849	**0.690**	**0.667**	0.714

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.859**	0.133	**1.000**	0.071
LR	0.649	0.273	0.375	0.214
RFsklearn	0.831	**0.629**	0.524	**0.786**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.5: Classification performance on the RUMC dataset of baseline models. Models were trained and hyperparameters were optimized on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.884**	0.862	**0.836**	0.889
LR	0.800	0.868	0.808	0.937
RFsklearn	0.848	**0.876**	0.811	**0.952**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	0.849	0.840	**0.893**	0.794
LR	**0.871**	0.836	0.864	0.810
RFsklearn	0.823	**0.852**	0.881	**0.825**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.6: Classification performance on the CWZ dataset of baseline models. Models were trained and hyperparameters were optimized on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

B.2 Cross-hospital evaluations

Figures B.7 and B.8 show evaluations of all baseline models trained on the Union/Liyuan hospital data and tested on the Liyuan/Union hospital data in the iCTCF dataset with mostly default parameters. Figures B.9 and B.10 show evaluations of all baseline models trained on the Union/Liyuan hospital data and tested on the Liyuan/Union hospital data in the iCTCF dataset with hyperparameter optimization.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.675	**0.797**	0.782	0.812
GBDT	**0.761**	0.606	**0.943**	0.446
LR	0.562	0.790	0.692	**0.919**
RFsklearn	0.656	0.774	0.774	0.774

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.798	**0.821**	0.903	0.753
GBDT	0.864	0.790	0.928	0.688
LR	0.800	0.800	0.794	**0.806**
RFsklearn	**0.873**	0.804	**0.936**	0.704

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.7: Classification performance on the iCTCF dataset of baseline models. Models were trained on data originating from the Union hospital and the evaluations shown here were computed based on data originating from the Liyuan hospital.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	**0.759**	0.867	0.766	**1.000**
GBDT	0.717	**0.868**	**0.772**	0.992
LR	0.489	0.756	0.768	0.743
RFsklearn	0.659	0.833	0.771	0.905

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.713	**0.868**	0.781	**0.978**
GBDT	0.730	0.852	0.775	0.945
LR	0.596	0.721	**0.791**	0.663
RFsklearn	**0.756**	0.866	0.781	0.972

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.8: Classification performance on the iCTCF dataset of baseline models. Models were trained on data originating from the Liyuan hospital and the evaluations shown here were computed based on data originating from the Union hospital.

Metric Model	AUC	F1	Precision	Recall
GBDT	0.577	**0.827**	0.705	**1.000**
LR	0.621	0.799	0.694	0.941
RFsklearn	**0.708**	0.801	**0.791**	0.812

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	0.848	0.777	0.932	0.667
LR	0.861	**0.856**	0.847	**0.866**
RFsklearn	**0.868**	0.811	**0.937**	0.715

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.9: Classification performance on the iCTCF dataset of baseline models. Models were trained and hyperparameters were optimized on data originating from the Union hospital and the evaluations shown here were computed based on data originating from the Liyuan hospital.

Figures B.11 and B.12 show evaluations of all baseline models trained on the RUMC/CWZ hospital data and tested on the CWZ/RUMC hospital data in the CWZ dataset with mostly default parameters Figures B.13 and B.14 show evaluations of all baseline models trained on the Union/Liyuan hospital data and tested on the Liyuan/Union hospital data in the iCTCF dataset with hyperparameter optimization.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.680**	**0.859**	0.752	**1.000**
LR	0.502	0.850	**0.772**	0.945
RFsklearn	0.671	0.855	0.768	0.964

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	0.734	0.866	0.786	0.964
LR	**0.777**	0.868	**0.801**	0.946
RFsklearn	0.752	**0.873**	0.793	**0.970**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.10: Classification performance on the iCTCF dataset of baseline models. Models were trained and hyperparameters were optimized on data originating from the Liyuan hospital and the evaluations shown here were computed based on data originating from the Union hospital.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.882	0.884	0.812	**0.969**
GBDT	0.858	0.880	0.829	0.939
LR	0.897	0.805	**0.958**	0.694
RFsklearn	**0.897**	**0.888**	0.834	0.949

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.854	0.856	0.846	0.867
GBDT	0.840	0.839	0.795	**0.888**
LR	**0.896**	0.802	**0.971**	0.684
RFsklearn	0.864	**0.860**	0.858	0.862

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.11: Classification performance on the CWZ dataset of baseline models. Models were trained on data originating from the RUMC hospital and the evaluations shown here were computed based on data originating from the CWZ hospital.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.736	0.814	0.686	**1.000**
GBDT	0.725	0.815	0.692	0.992
LR	0.703	0.765	0.692	0.856
RFsklearn	**0.769**	**0.822**	**0.698**	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.823	0.796	0.709	0.907
GBDT	0.807	**0.832**	0.720	**0.983**
LR	0.810	0.790	**0.754**	0.831
RFsklearn	**0.833**	0.816	0.732	0.924

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.12: Classification performance on the CWZ dataset of baseline models. Models were trained on data originating from the CWZ hospital and the evaluations shown here were computed based on data originating from the RUMC hospital.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.886**	0.856	0.748	**1.000**
LR	0.796	0.856	0.748	**1.000**
RFsklearn	0.868	**0.856**	**0.819**	0.898

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	0.838	0.856	0.748	**1.000**
LR	**0.890**	**0.887**	**0.918**	0.857
RFsklearn	0.842	0.850	0.863	0.837

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.13: Classification performance on the CWZ dataset of baseline models. Models were trained and hyperparameters were optimized on data originating from the RUMC hospital and the evaluations shown here were computed based on data originating from the CWZ hospital.

Metric Model	AUC	F1	Precision	Recall
GBDT	0.751	**0.814**	0.686	**1.000**
LR	**0.779**	0.812	**0.688**	0.992
RFsklearn	0.755	**0.814**	0.686	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	0.808	0.814	0.686	**1.000**
LR	**0.889**	**0.853**	**0.769**	0.958
RFsklearn	0.834	0.814	0.686	**1.000**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.14: Classification performance on the CWZ dataset of baseline models. Models were trained and hyperparameters were optimized on data originating from the CWZ hospital and the evaluations shown here were computed based on data originating from the RUMC hospital.

B.3 Evaluating the effects of missing data

Figures B.15, B.16, and B.17 show the performance of all of the baseline models when predicting on missingness labels (true when a value is present and false when it is missing) for the iCTCF, RUMC, and CWZ datasets respectively.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	**0.954**	0.887	0.837	0.943
GBDT	0.932	0.907	0.866	0.951
LR	0.943	0.892	0.844	0.947
RFsklearn	0.945	**0.911**	**0.870**	**0.955**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.955	0.894	0.849	0.943
GBDT	0.942	0.914	0.879	0.951
LR	0.944	0.894	0.849	0.943
RFsklearn	**0.956**	**0.932**	**0.911**	**0.955**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.15: Classification performance on the iCTCF dataset of baseline models. Models were trained on missingness labels from the first cohort and the evaluations shown here were computed based on missingness labels from the second cohort.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	**0.879**	**0.710**	0.647	**0.786**
GBDT	0.863	0.690	0.667	0.714
LR	0.853	0.583	**0.700**	0.500
RFsklearn	0.799	0.125	0.500	0.071

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	**0.885**	**0.710**	0.647	**0.786**
GBDT	0.863	0.690	0.667	0.714
LR	0.853	0.583	**0.700**	0.500
RFsklearn	0.784	0.125	0.500	0.071

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.16: Classification performance on the RUMC dataset of baseline models. Models were trained on missingness labels from the predefined train/val set and the evaluations shown here were computed based on missingness labels from the predefined test set.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.667	**0.829**	**0.753**	**0.921**
GBDT	**0.702**	**0.829**	**0.753**	**0.921**
LR	0.634	0.814	0.740	0.905
RFsklearn	0.654	0.785	0.736	0.841

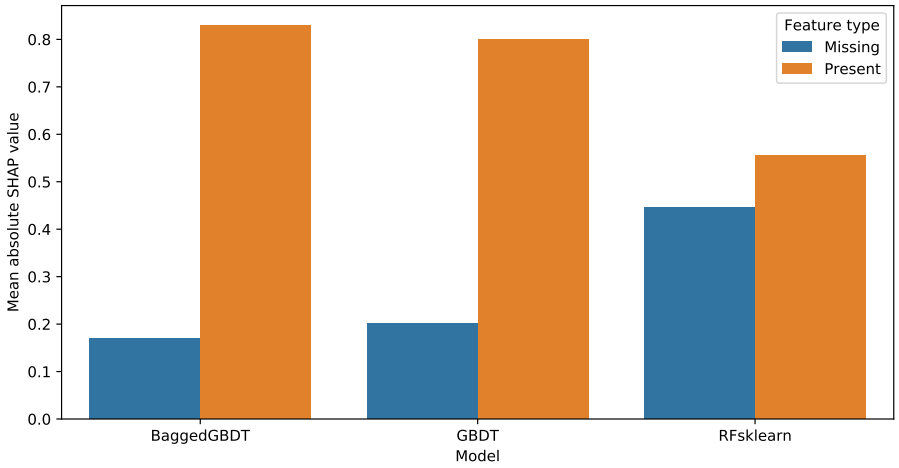
(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.666	**0.829**	**0.753**	**0.921**
GBDT	**0.702**	**0.829**	**0.753**	**0.921**
LR	0.633	0.803	0.722	0.905
RFsklearn	0.661	0.800	0.750	0.857

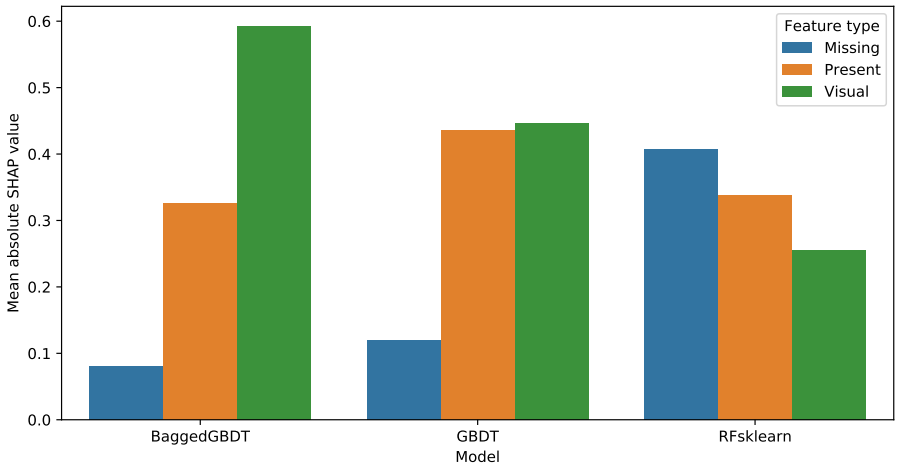
(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.17: Classification performance on the CWZ dataset of baseline models. Models were trained on missingness labels from the predefined train/val set and the evaluations shown here were computed based on missingness labels from the predefined test set.

The last evaluation of the extent to which models fit on missingness is that of the average absolute Shapley values. Figures B.1, B.2, and B.3 show the average absolute SHAP values of each of the baseline models for all missing features, present features, and visual features (where applicable).



(A) Model performance based on clinical features alone.

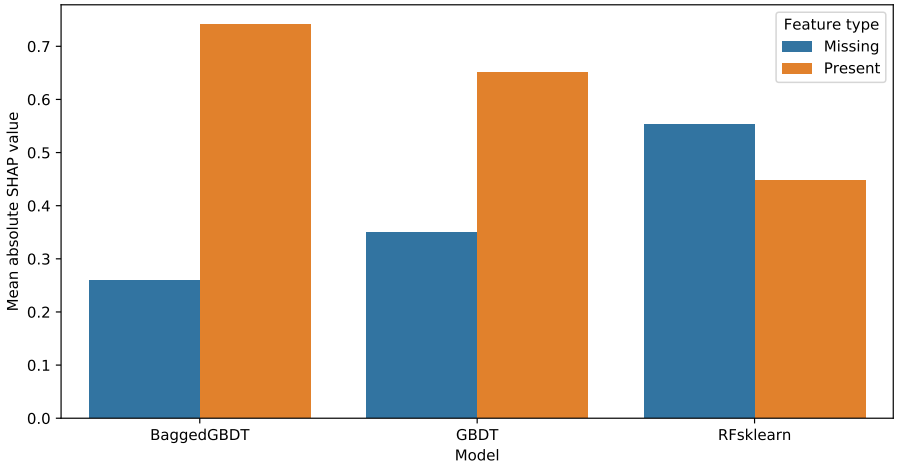


(B) Model performance based on both clinical features and visual (CT) features.

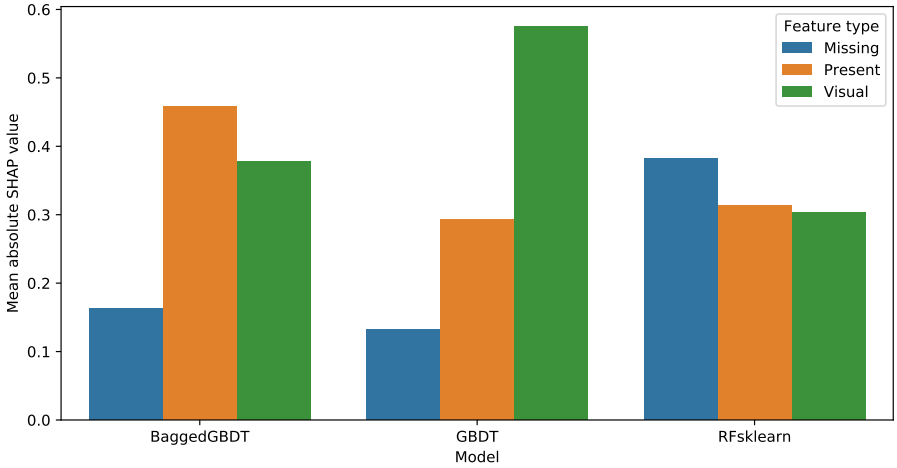
FIGURE B.1: Mean absolute Shapley values when trained on the iCTCF dataset of baseline models. Models were trained on data from the first cohort and the evaluations shown here were computed based on data from the second cohort.

B.4 Preventing fitting on missingness

This final section shows the results of the various method we implemented to try to reduce the amount of fitting on missingness. They are firstly evaluated on a toy dataset (specifically the breast cancer dataset distributed with scikit-learn), which was repeated synthetically amputated in order to simulate problematic missingness patterns.



(A) Model performance based on clinical features alone.



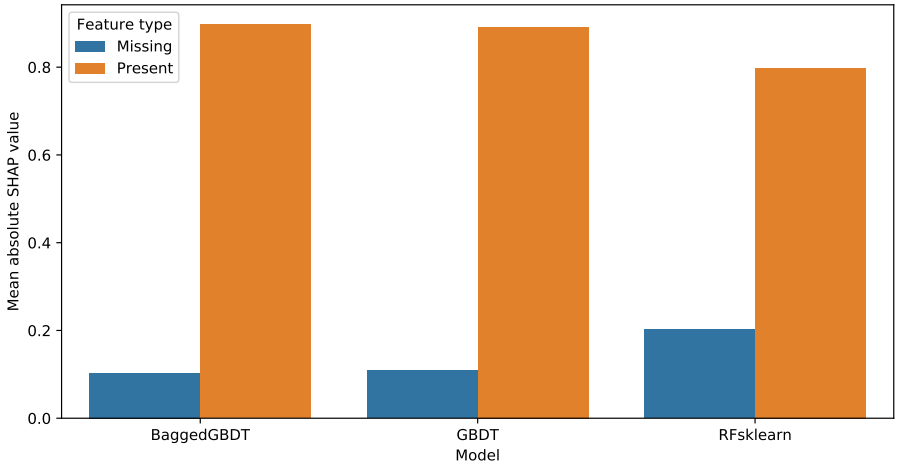
(B) Model performance based on both clinical features and visual (CT) features.

FIGURE B.2: Mean absolute Shapley values when trained on the RUMC dataset of baseline models. Models were trained on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

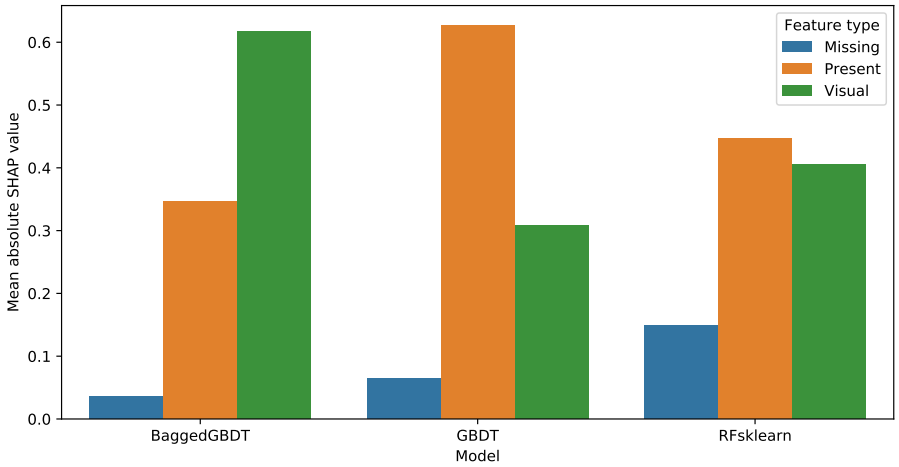
Figure B.4 shows the performance of a baseline GBDT model, mean imputation, 5-Nearest Neighbours imputation (KNN5), MICE imputation (MICE), a custom decision tree implementation (DT), SHAP-zeroing using a GBDT model, an ensemble multiple imputation model (MI), and an ensembled multiple imputation model using test-time imputations (MI TT).

Based on the accuracy distributions for each model, we ran paired one-sided t-tests with Bonferonni correction. The results of this can be seen in table B.18. Table B.19 shows a ranking of the models based on the p-values (< 0.01 is considered significant).

Tables B.20, B.21, B.22, and B.23 show the performance of all of the baselines on the iCTCF and CWZ datasets, using various imputation strategies. Both datasets are tested by using cross-hospital train/test splits. In addition, the default train/test splits were also evaluated and are shown in tables B.24, B.25, and B.26.



(A) Model performance based on clinical features alone.



(B) Model performance based on both clinical features and visual (CT) features.

FIGURE B.3: Mean absolute Shapley values when trained on the CWZ dataset of baseline models. Models were trained on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

Tables B.27, B.28, B.29, and B.30 show the performance of all of the baselines on the iCTCF and CWZ datasets, using SHAP-zeroing. Both datasets are tested by using cross-hospital train/test splits. In addition, the default train/test splits were also evaluated and are shown in tables B.31, B.32, and B.33.

Tables B.34, B.35, B.36, and B.37 show the performance of all of the baselines on the iCTCF and CWZ datasets using SHAP-zeroing and hyperparameter optimization. Both datasets are tested by using cross-hospital train/test splits. In addition, the default train/test splits were also evaluated and are shown in tables B.38, B.39, and B.40.

Tables B.41, B.42, B.43, and B.44 show the performance of all of the baselines on the iCTCF and CWZ datasets using multiple imputation ensembles. Both datasets are tested by using cross-hospital train/test splits. In addition, the default train/test splits were also evaluated and are shown in tables B.45, B.46, and B.47.

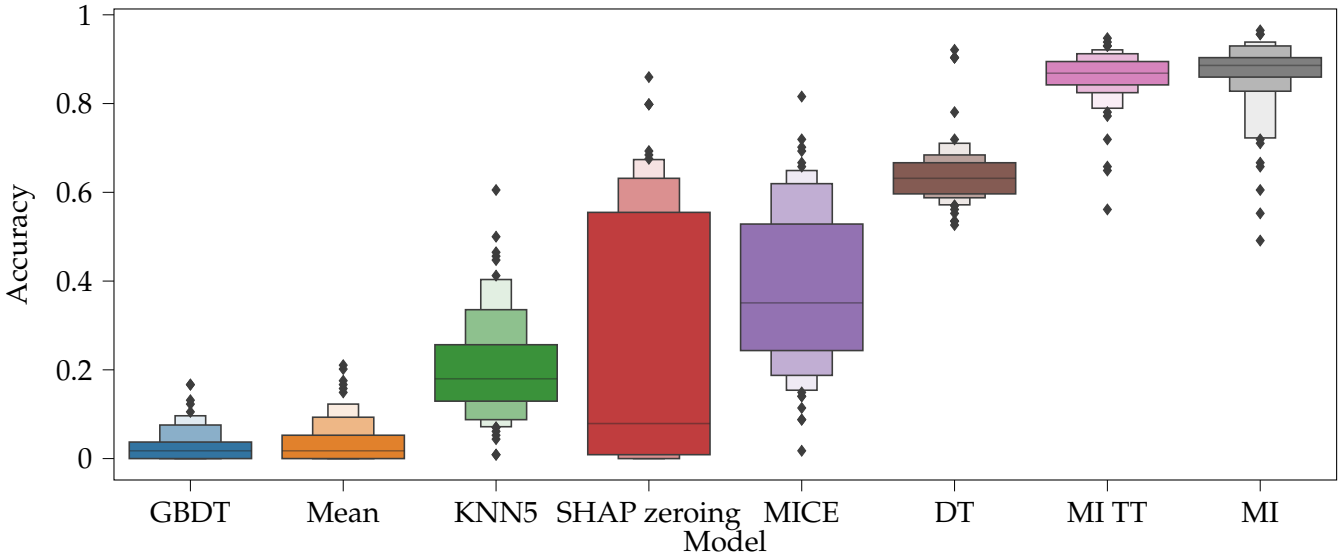


FIGURE B.4: Accuracy scores of 100 random augmented datasets for various models.

TABLE B.18: P-values of one-sided paired t-test between the accuracies of all models evaluated on 100 random augmented datasets.

Second First	DT	GBDT	KNN5	MI	MI TT	MICE	Mean	SHAP zeroing
DT	-	0.000	0.000	1.000	1.000	0.000	0.000	0.000
GBDT	1.000	-	1.000	1.000	1.000	1.000	0.943	1.000
KNN5	1.000	0.000	-	1.000	1.000	1.000	0.000	0.818
MI	0.000	0.000	0.000	-	0.154	0.000	0.000	0.000
MI TT	0.000	0.000	0.000	0.846	-	0.000	0.000	0.000
MICE	1.000	0.000	0.000	1.000	1.000	-	0.000	0.000
Mean	1.000	0.057	1.000	1.000	1.000	1.000	-	1.000
SHAP zeroing	1.000	0.000	0.182	1.000	1.000	1.000	0.000	-

TABLE B.19: Ranking of models in synthetic data.

Rank	Model
1	MI
1	MI TT
2	DT
3	MICE
4	SHAP zeroing
4	KNN5
5	GBDT
5	Mean

Model	Metric Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	0.687	**0.818**	0.709	**0.968**
	MEAN	**0.724**	0.775	0.755	0.796
GBDT	KNN5	0.666	0.789	0.725	0.866
	MEAN	0.713	0.654	**0.833**	0.538
	MICE	0.519	0.778	0.733	0.828
LR	KNN5	0.634	0.798	0.708	0.914
	MEAN	0.562	0.790	0.692	0.919
	MICE	0.580	0.817	0.718	0.946
RFsklearn	KNN5	0.620	0.802	0.708	0.925
	MEAN	0.680	0.777	0.797	0.758
	MICE	0.476	0.742	0.707	0.780

(A) Model performance based on clinical features alone.

Model	Metric Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	0.862	0.838	0.790	**0.892**
	MEAN	0.873	0.832	0.939	0.747
GBDT	KNN5	0.842	0.850	0.879	0.823
	MEAN	0.875	0.806	**0.942**	0.704
	MICE	**0.879**	0.849	0.924	0.785
LR	KNN5	0.811	0.821	0.782	0.866
	MEAN	0.800	0.800	0.794	0.806
	MICE	0.846	0.840	0.821	0.860
RFsklearn	KNN5	0.871	0.851	0.914	0.796
	MEAN	0.871	0.807	0.936	0.710
	MICE	0.853	**0.857**	0.936	0.790

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.20: Classification performance on the iCTCF dataset of .42. Models were trained on imputed data originating from the Union hospital and the evaluations shown here were computed based on data originating from the Liyuan hospital.

Model	Metric Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	0.688	0.858	0.757	0.991
	MEAN	0.682	**0.866**	0.767	**0.996**
GBDT	KNN5	0.613	0.855	0.760	0.979
	MEAN	0.640	0.864	0.772	0.980
	MICE	0.624	0.822	0.768	0.884
LR	KNN5	0.517	0.802	**0.777**	0.827
	MEAN	0.489	0.756	0.768	0.743
	MICE	0.488	0.806	0.773	0.842
RFsklearn	KNN5	**0.702**	0.858	0.764	0.979
	MEAN	0.655	0.828	0.771	0.893
	MICE	0.676	0.846	0.761	0.954

(A) Model performance based on clinical features alone.

Model	Metric Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	**0.811**	0.865	0.781	0.968
	MEAN	0.790	**0.869**	0.783	**0.975**
GBDT	KNN5	0.772	0.863	0.778	0.970
	MEAN	0.787	0.864	0.782	0.964
	MICE	0.726	0.842	0.768	0.931
LR	KNN5	0.513	0.813	0.772	0.859
	MEAN	0.596	0.721	**0.791**	0.663
	MICE	0.552	0.836	0.786	0.893
RFsklearn	KNN5	0.796	0.863	0.775	0.972
	MEAN	0.748	0.866	0.780	0.972
	MICE	0.704	0.859	0.777	0.962

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.21: Classification performance on the iCTCF dataset of .4. Models were trained on imputed data originating from the Liyuan hospital and the evaluations shown here were computed based on data originating from the Union hospital.

Model	Metric / Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	0.888	0.872	0.792	0.969
	MEAN	0.878	0.875	0.827	0.929
GBDT	KNN5	0.842	0.863	0.827	0.903
	MEAN	0.862	0.872	0.872	0.872
	MICE	0.873	0.872	0.843	0.903
LR	KNN5	0.854	0.863	0.844	0.883
	MEAN	**0.897**	0.805	**0.958**	0.694
	MICE	0.774	0.850	0.748	**0.985**
RFsklearn	KNN5	0.875	0.874	0.795	0.969
	MEAN	0.896	**0.885**	0.833	0.944
	MICE	0.876	0.860	0.780	0.959

(A) Model performance based on clinical features alone.

Model	Metric / Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	0.856	**0.860**	0.858	0.862
	MEAN	0.856	0.859	0.862	0.857
GBDT	KNN5	0.846	0.852	0.814	0.893
	MEAN	0.857	0.854	0.831	0.878
	MICE	0.855	0.857	0.842	0.872
LR	KNN5	0.741	0.826	0.757	**0.908**
	MEAN	**0.896**	0.802	**0.971**	0.684
	MICE	0.841	0.859	0.822	0.898
RFsklearn	KNN5	0.856	0.853	0.848	0.857
	MEAN	0.860	0.857	0.857	0.857
	MICE	0.853	0.855	0.853	0.857

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.22: Classification performance on the CWZ dataset of .4. Models were trained on data originating from the RUMC hospital and the evaluations shown here were computed based on imputed data originating from the CWZ hospital.

Model	Metric Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	0.773	0.819	0.706	0.975
	MEAN	0.736	0.814	0.686	**1.000**
GBDT	KNN5	0.743	**0.834**	**0.739**	0.958
	MEAN	0.697	0.812	0.688	0.992
	MICE	0.694	0.824	0.705	0.992
LR	KNN5	0.696	0.769	0.704	0.847
	MEAN	0.703	0.765	0.692	0.856
	MICE	0.758	0.809	0.725	0.915
RFsklearn	KNN5	**0.776**	0.832	0.731	0.966
	MEAN	0.762	0.817	0.690	**1.000**
	MICE	0.774	0.821	0.701	0.992

(A) Model performance based on clinical features alone.

Model	Metric Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	0.851	0.812	0.754	0.881
	MEAN	0.836	0.818	0.696	0.992
GBDT	KNN5	0.848	0.849	0.780	0.932
	MEAN	0.845	0.827	0.709	0.992
	MICE	0.826	0.843	0.728	**1.000**
LR	KNN5	0.836	0.841	0.791	0.898
	MEAN	0.810	0.790	0.754	0.831
	MICE	**0.859**	**0.856**	**0.811**	0.907
RFsklearn	KNN5	0.836	0.824	0.766	0.890
	MEAN	0.829	0.817	0.743	0.907
	MICE	0.834	0.812	0.719	0.932

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.23: Classification performance on the CWZ dataset of .4. Models were trained on imputed data originating from the CWZ hospital and the evaluations shown here were computed based on imputed data originating from the RUMC hospital.

Model	Metric / Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	0.781	0.835	0.739	0.959
	MEAN	0.881	0.891	0.814	**0.984**
GBDT	KNN5	0.823	0.856	0.792	0.931
	MEAN	**0.925**	**0.904**	**0.837**	**0.984**
	MICE	0.773	0.823	0.743	0.922
LR	KNN5	0.639	0.779	0.755	0.804
	MEAN	0.617	0.744	0.741	0.747
	MICE	0.602	0.774	0.753	0.796
RFsklearn	KNN5	0.824	0.836	0.741	0.959
	MEAN	0.910	0.863	0.772	0.980
	MICE	0.791	0.826	0.725	0.959

(A) Model performance based on clinical features alone.

Model	Metric / Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	0.821	0.810	0.707	0.947
	MEAN	0.870	0.825	0.722	**0.963**
GBDT	KNN5	0.854	0.822	0.729	0.943
	MEAN	**0.895**	**0.842**	**0.752**	0.955
	MICE	0.836	0.821	0.725	0.947
LR	KNN5	0.636	0.785	0.724	0.857
	MEAN	0.689	0.756	0.741	0.771
	MICE	0.613	0.782	0.741	0.829
RFsklearn	KNN5	0.814	0.804	0.708	0.931
	MEAN	0.836	0.816	0.725	0.935
	MICE	0.822	0.812	0.718	0.935

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.24: Classification performance on the iCTCF dataset of baseline models. Models were trained on imputed data from the first cohort and the evaluations shown here were computed based on imputed data from the second cohort.

Model	Metric Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	0.831	0.400	**0.667**	0.286
	MEAN	**0.833**	0.444	0.462	0.429
GBDT	KNN5	0.792	0.480	0.545	0.429
	MEAN	0.794	**0.533**	0.500	**0.571**
	MICE	0.754	0.519	0.538	0.500
LR	KNN5	0.629	0.467	0.438	0.500
	MEAN	0.601	0.417	0.500	0.357
	MICE	0.661	0.333	0.400	0.286
RFsklearn	KNN5	0.732	0.105	0.200	0.071
	MEAN	0.731	0.200	0.333	0.143
	MICE	0.774	0.125	0.500	0.071

(A) Model performance based on clinical features alone.

Model	Metric Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	**0.831**	0.316	0.600	0.214
	MEAN	0.815	**0.538**	0.583	**0.500**
GBDT	KNN5	0.732	nan	nan	nan
	MEAN	0.782	0.348	0.444	0.286
	MICE	0.790	0.333	**0.750**	0.214
LR	KNN5	0.583	0.500	0.600	0.429
	MEAN	0.657	0.364	0.500	0.286
	MICE	0.671	0.455	0.625	0.357
RFsklearn	KNN5	0.743	nan	nan	nan
	MEAN	0.782	nan	nan	nan
	MICE	0.786	nan	nan	nan

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.25: Classification performance on the RUMC dataset of baseline models. Models were trained on an imputed version of the predefined train/val set and the evaluations shown here were computed based on an imputed version of the predefined test set.

Model	Metric Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	0.881	0.872	0.829	0.921
	MEAN	0.862	0.872	0.829	0.921
GBDT	KNN5	0.858	0.875	**0.862**	0.889
	MEAN	0.862	0.877	0.851	0.905
	MICE	0.870	0.877	0.851	0.905
LR	KNN5	0.807	0.840	0.809	0.873
	MEAN	0.798	0.846	0.821	0.873
	MICE	0.809	0.828	0.815	0.841
RFsklearn	KNN5	**0.887**	**0.885**	0.853	0.921
	MEAN	0.877	0.881	0.831	**0.937**
	MICE	0.865	0.874	0.819	**0.937**

(A) Model performance based on clinical features alone.

Model	Metric Impute method	AUC	F1	Precision	Recall
BaggedGBDT	KNN5	0.819	0.829	0.850	0.810
	MEAN	0.804	0.820	0.847	0.794
GBDT	KNN5	0.844	**0.855**	0.869	0.841
	MEAN	0.836	0.848	0.855	0.841
	MICE	0.852	0.848	0.855	0.841
LR	KNN5	**0.870**	0.817	0.860	0.778
	MEAN	0.857	0.840	**0.893**	0.794
	MICE	0.788	0.853	0.833	**0.873**
RFsklearn	KNN5	0.823	0.826	0.862	0.794
	MEAN	0.823	0.820	0.847	0.794
	MICE	0.844	0.836	0.864	0.810

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.26: Classification performance on the CWZ dataset of baseline models. Models were trained on an imputed version of the predefined train/val set and the evaluations shown here were computed based on an imputed version of the predefined test set.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.637	0.799	0.694	0.941
GBDT	0.644	0.795	0.701	0.919
LR	0.562	0.790	0.692	0.919
RFsklearn	**0.655**	**0.827**	**0.705**	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.854	**0.851**	**0.893**	0.812
GBDT	0.849	0.848	0.874	0.823
LR	0.800	0.800	0.794	0.806
RFsklearn	**0.865**	0.827	0.705	**1.000**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.27: Classification performance on the iCTCF dataset of SHAP zeroed models. Models were trained on data originating from the Union hospital and the evaluations shown here were computed based on data originating from the Liyuan hospital.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	**0.753**	0.861	0.755	**1.000**
GBDT	0.711	**0.864**	0.764	0.996
LR	0.534	0.756	**0.769**	0.743
RFsklearn	0.652	0.859	0.752	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.731	**0.871**	0.783	0.980
GBDT	**0.777**	0.866	0.778	0.978
LR	0.567	0.722	**0.792**	0.663
RFsklearn	0.747	0.859	0.752	**1.000**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.28: Classification performance on the iCTCF dataset of SHAP zeroed models. Models were trained on data originating from the Liyuan hospital and the evaluations shown here were computed based on data originating from the Union hospital.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.879	0.882	0.809	0.969
GBDT	0.860	**0.882**	0.852	0.913
LR	**0.897**	0.805	**0.958**	0.694
RFsklearn	0.891	0.856	0.748	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.852	**0.860**	0.858	0.862
GBDT	0.846	0.842	0.817	0.867
LR	**0.896**	0.802	**0.971**	0.684
RFsklearn	0.867	0.856	0.748	**1.000**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.29: Classification performance on the CWZ dataset of SHAP zeroed models. Models were trained on data originating from the RUMC hospital and the evaluations shown here were computed based on data originating from the CWZ hospital.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	**0.777**	0.814	0.686	**1.000**
GBDT	0.776	**0.819**	**0.706**	0.975
LR	0.703	0.765	0.692	0.856
RFsklearn	0.750	0.814	0.686	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.836	0.797	0.716	0.898
GBDT	**0.860**	**0.821**	0.733	0.932
LR	0.810	0.790	**0.754**	0.831
RFsklearn	0.834	0.814	0.686	**1.000**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.30: Classification performance on the CWZ dataset of SHAP zeroed models. Models were trained on data originating from the CWZ hospital and the evaluations shown here were computed based on data originating from the RUMC hospital.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.871	**0.879**	**0.799**	0.976
GBDT	0.908	0.865	0.782	0.967
LR	0.617	0.744	0.741	0.747
RFsklearn	**0.919**	0.822	0.698	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.855	0.822	0.719	0.959
GBDT	**0.861**	**0.841**	**0.747**	0.963
LR	0.689	0.756	0.741	0.771
RFsklearn	0.838	0.822	0.698	**1.000**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.31: Classification performance on the iCTCF dataset of SHAP zeroed models. Models were trained on data from the first cohort and the evaluations shown here were computed based on data from the second cohort.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	**0.901**	**0.741**	**0.769**	0.714
GBDT	0.883	0.634	0.481	**0.929**
LR	0.601	0.417	0.500	0.357
RFsklearn	0.743	nan	nan	nan

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	**0.889**	**0.649**	0.522	**0.857**
GBDT	0.843	0.640	**0.727**	0.571
LR	0.657	0.364	0.500	0.286
RFsklearn	0.786	nan	nan	nan

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.32: Classification performance on the RUMC dataset of SHAP zeroed models. Models were trained on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.876	0.863	0.789	0.952
GBDT	0.862	**0.864**	**0.826**	0.905
LR	0.798	0.846	0.821	0.873
RFsklearn	**0.876**	0.840	0.724	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
BaggedGBDT	0.807	0.829	0.850	0.810
GBDT	0.841	0.839	0.852	0.825
LR	**0.857**	**0.840**	**0.893**	0.794
RFsklearn	0.821	0.840	0.724	**1.000**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.33: Classification performance on the CWZ dataset of SHAP zeroed models. Models were trained on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

Metric Model	AUC	F1	Precision	Recall
GBDT	0.525	**0.827**	**0.705**	**1.000**
RFsklearn	**0.682**	**0.827**	**0.705**	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.889**	**0.887**	**0.935**	0.844
RFsklearn	0.878	0.827	0.705	**1.000**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.34: Classification performance on the iCTCF dataset of SHAP zeroed models. Models were trained and hyperparameters were optimized on data originating from the Union hospital and the evaluations shown here were computed based on data originating from the Liyuan hospital.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.670**	**0.859**	**0.752**	**1.000**
RFsklearn	0.630	**0.859**	**0.752**	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.760**	**0.859**	**0.752**	**1.000**
RFsklearn	0.757	**0.859**	**0.752**	**1.000**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.35: Classification performance on the iCTCF dataset of SHAP zeroed models. Models were trained and hyperparameters were optimized on imputed data originating from the Liyuan hospital and the evaluations shown here were computed based on data originating from the Union hospital.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.893**	**0.856**	**0.748**	**1.000**
RFsklearn	0.890	**0.856**	**0.748**	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.863**	**0.856**	**0.748**	**1.000**
RFsklearn	0.855	**0.856**	**0.748**	**1.000**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.36: Classification performance on the CWZ dataset of SHAP zeroed models. Models were trained and hyperparameters were optimized on data originating from the RUMC hospital and the evaluations shown here were computed based on imputed data originating from the CWZ hospital.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.809**	**0.814**	**0.686**	**1.000**
RFsklearn	0.772	**0.814**	**0.686**	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.867**	**0.815**	**0.724**	0.932
RFsklearn	0.828	0.814	0.686	**1.000**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.37: Classification performance on the CWZ dataset of SHAP zeroed models. Models were trained and hyperparameters were optimized on imputed data originating from the CWZ hospital and the evaluations shown here were computed based on imputed data originating from the RUMC hospital.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.939**	**0.822**	**0.698**	**1.000**
RFsklearn	0.904	**0.822**	**0.698**	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	0.812	**0.822**	**0.698**	**1.000**
RFsklearn	**0.843**	**0.822**	**0.698**	**1.000**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.38: Classification performance on the iCTCF dataset of SHAP zeroed models. Models were trained and hyperparameters were optimized on data from the first cohort and the evaluations shown here were computed based on data from the second cohort.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.889**	**0.710**	**0.647**	**0.786**
RFsklearn	0.759	0.105	0.200	0.071

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.865**	**0.600**	**0.462**	**0.857**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.39: Classification performance on the RUMC dataset of SHAP zeroed models. Models were trained and hyperparameters were optimized on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.868**	**0.840**	**0.724**	**1.000**
RFsklearn	0.849	**0.840**	**0.724**	**1.000**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.874**	**0.857**	**0.857**	0.857
RFsklearn	0.839	0.840	0.724	**1.000**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.40: Classification performance on the CWZ dataset of SHAP zeroed models. Models were trained and hyperparameters were optimized on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

Metric Model	AUC	F1	Precision	Recall
GBDT	0.643	0.781	0.695	0.892
LR	**0.657**	**0.808**	**0.696**	**0.962**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.863**	**0.857**	**0.936**	0.790
LR	0.834	0.854	0.828	**0.882**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.41: Classification performance on the iCTCF dataset of multiple imputation ensemble models. Models were trained on imputed data originating from the Union hospital and the evaluations shown here were computed based on data originating from the Liyuan hospital.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.667**	**0.854**	0.763	**0.968**
LR	0.607	0.830	**0.769**	0.903

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.868**	**0.870**	0.790	**0.967**
LR	0.760	0.857	**0.800**	0.924

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.42: Classification performance on the iCTCF dataset of multiple imputation ensemble models. Models were trained on imputed data originating from the Liyuan hospital and the evaluations shown here were computed based on data originating from the Union hospital.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.874**	**0.872**	**0.861**	0.883
LR	0.850	0.870	0.836	**0.908**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.854**	**0.855**	0.853	**0.857**
LR	0.840	0.850	**0.876**	0.827

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.43: Classification performance on the CWZ dataset of multiple imputation ensemble models. Models were trained on data originating from the RUMC hospital and the evaluations shown here were computed based on imputed data originating from the CWZ hospital.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.756**	**0.811**	0.690	**0.983**
LR	0.705	0.770	**0.694**	0.864

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.840**	**0.819**	0.725	**0.941**
LR	0.835	0.811	**0.786**	0.839

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.44: Classification performance on the CWZ dataset of multiple imputation ensemble models. Models were trained on imputed data originating from the CWZ hospital and the evaluations shown here were computed based on imputed data originating from the RUMC hospital.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.760**	**0.806**	**0.754**	**0.865**
LR	0.729	0.803	0.749	**0.865**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.799**	**0.811**	0.726	**0.918**
LR	0.758	0.802	**0.733**	0.886

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.45: Classification performance on the iCTCF dataset of multiple imputation ensemble models. Models were trained on data from the first cohort and the evaluations shown here were computed based on data from the second cohort.

Metric Model	AUC	F1	Precision	Recall
GBDT	0.780	**0.571**	**0.571**	**0.571**
LR	**0.808**	0.480	0.545	0.429

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.770**	**0.621**	**0.600**	**0.643**
LR	0.752	0.500	0.500	0.500

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.46: Classification performance on the RUMC dataset of multiple imputation ensemble models. Models were trained on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

Metric Model	AUC	F1	Precision	Recall
GBDT	**0.827**	0.846	0.821	0.873
LR	0.795	**0.864**	**0.826**	**0.905**

(A) Model performance based on clinical features alone.

Metric Model	AUC	F1	Precision	Recall
GBDT	0.806	0.783	0.825	0.746
LR	**0.839**	**0.810**	**0.845**	**0.778**

(B) Model performance based on both clinical features and visual (CT) features.

TABLE B.47: Classification performance on the CWZ dataset of multiple imputation ensemble models. Models were trained on the predefined train/val set and the evaluations shown here were computed based on the predefined test set.

Bibliography

Arevalo-Rodriguez, Ingrid et al. (2020). "False-negative results of initial RT-PCR assays for COVID-19: a systematic review". In: *PloS one* 15.12, e0242958.

Bai, Harrison X et al. (2020). "Artificial intelligence augmentation of radiologist performance in distinguishing COVID-19 from pneumonia of other origin at chest CT". In: *Radiology* 296.3, E156–E165.

Barrasa, Helena et al. (2020). "SARS-CoV-2 in Spanish intensive care units: early experience with 15-day survival in Vitoria". In: *Anaesthesia Critical Care & Pain Medicine* 39.5, pp. 553–561.

Baskin, Chaim et al. (2018). "Streaming architecture for large-scale quantized neural networks on an FPGA-based dataflow platform". In: *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, pp. 162–169.

Bergstra, James and Yoshua Bengio (2012). "Random search for hyper-parameter optimization." In: *Journal of machine learning research* 13.2.

Bergstra, James et al. (2011). "Algorithms for hyper-parameter optimization". In: *Advances in neural information processing systems* 24.

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

— (1999). "Random forests". In: *UC Berkeley TR567*.

Breiman, Leo et al. (2017). *Classification and regression trees*. Routledge.

Buuren, S van and Karin Groothuis-Oudshoorn (2010). "mice: Multivariate imputation by chained equations in R". In: *Journal of statistical software*, pp. 1–68.

Carreira, Joao and Andrew Zisserman (2017). "Quo vadis, action recognition? a new model and the kinetics dataset". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308.

Chan, Jasper Fuk-Woo et al. (2020). "Improved molecular diagnosis of COVID-19 by the novel, highly sensitive and specific COVID-19-RdRp/Hel real-time reverse transcription-PCR assay validated in vitro and with clinical specimens". In: *Journal of clinical microbiology* 58.5.

Cnudde, Veerle and Matthieu Nicolaas Boone (2013). "High-resolution X-ray computed tomography in geosciences: A review of the current technology and applications". In: *Earth-Science Reviews* 123, pp. 1–17.

Crozier, Alex et al. (2021). "Put to the test: use of rapid testing technologies for covid-19". In: *bmj* 372.

Donders, A Rogier T et al. (2006). "A gentle introduction to imputation of missing values". In: *Journal of clinical epidemiology* 59.10, pp. 1087–1091.

Efron, Bradley and Robert Tibshirani (1997). "Improvements on cross-validation: the 632+ bootstrap method". In: *Journal of the American Statistical Association* 92.438, pp. 548–560.

Emmert-Streib, Frank et al. (2020). "An introductory review of deep learning for prediction models with big data". In: *Frontiers in Artificial Intelligence* 3, p. 4.

Freund, Yoav, Robert Schapire, and Naoki Abe (1999). "A short introduction to boosting". In: *Journal-Japanese Society For Artificial Intelligence* 14.771-780, p. 1612.

Gralinski, Lisa E and Vineet D Menachery (2020). "Return of the Coronavirus: 2019-nCoV". In: *Viruses* 12.2, p. 135.

Guelman, Leo (2012). "Gradient boosting trees for auto insurance loss cost modeling and prediction". In: *Expert Systems with Applications* 39.3, pp. 3659–3667.

Han, Huanqin et al. (2020). "SARS-CoV-2 RNA more readily detected in induced sputum than in throat swabs of convalescent COVID-19 patients". In: *The Lancet Infectious Diseases* 20.6, pp. 655–656.

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

Hu, Ben et al. (2020). "Characteristics of SARS-CoV-2 and COVID-19". In: *Nature Reviews Microbiology*, pp. 1–14.

Hui, David S et al. (2020). "The continuing 2019-nCoV epidemic threat of novel coronaviruses to global health—The latest 2019 novel coronavirus outbreak in Wuhan, China". In: *International journal of infectious diseases* 91, pp. 264–266.

Jin, Cheng et al. (2020). "Development and evaluation of an artificial intelligence system for COVID-19 diagnosis". In: *Nature communications* 11.1, pp. 1–14.

Ke, Guolin et al. (2017). "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems* 30, pp. 3146–3154.

Konrad, Regina et al. (2020). "Rapid establishment of laboratory diagnostics for the novel coronavirus SARS-CoV-2 in Bavaria, Germany, February 2020". In: *Eurosurveillance* 25.9, p. 2000173.

Kwee, Thomas C and Robert M Kwee (2020). "Chest CT in COVID-19: what the radiologist needs to know". In: *RadioGraphics* 40.7, pp. 1848–1865.

Lessmann, Nikolas et al. (2020). "Automated assessment of CO-RADS and chest CT severity scores in patients with suspected COVID-19 using artificial intelligence". In: *Radiology*.

Li, Lin et al. (2020). "Artificial intelligence distinguishes COVID-19 from community acquired pneumonia on chest CT". In: *Radiology*.

Lippi, Giuseppe, Brandon M Henry, and Fabian Sanchis-Gomar (2021). "Red blood cell distribution is a significant predictor of severe illness in coronavirus disease 2019". In: *Acta Haematologica* 144.4, pp. 4–8.

Lu, Renfei et al. (2020). "Development of a novel reverse transcription loop-mediated isothermal amplification method for rapid detection of SARS-CoV-2". In: *Virologica Sinica* 35.3, pp. 344–347.

Lundberg, Scott M and Su-In Lee (2017). "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

Lundberg, Scott M et al. (2020a). "From local explanations to global understanding with explainable AI for trees". In: *Nature machine intelligence* 2.1, pp. 56–67.

Lundberg, Scott M. et al. (2020b). "From local explanations to global understanding with explainable AI for trees". In: *Nature Machine Intelligence* 2.1, pp. 2522–5839.

Mason, Llew et al. (1999). "Boosting algorithms as gradient descent in function space". In: Nips.

Molnar, Christoph (2020). *Interpretable machine learning*. Lulu. com.

Ning, Wanshan et al. (2020). "iCTCF: an integrative resource of chest computed tomography images and clinical features of patients with COVID-19 pneumonia". In:

Pan, Yang et al. (2020). "Viral load of SARS-CoV-2 in clinical samples". In: *The Lancet infectious diseases* 20.4, pp. 411–412.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Prokop, Mathias et al. (2020). "CO-RADS: a categorical CT assessment scheme for patients suspected of having COVID-19—definition and evaluation". In: *Radiology* 296.2, E97–E104.

Rokach, Lior and Oded Maimon (2005). "Decision trees". In: *Data mining and knowledge discovery handbook*. Springer, pp. 165–192.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.

Rosenblatt, Frank (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

Rubin, Donald B (1976). "Inference and missing data". In: *Biometrika* 63.3, pp. 581–592.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Saberian, Mohammad, Pablo Delgado, and Yves Raimond (2019). "Gradient boosted decision tree neural network". In: *arXiv preprint arXiv:1910.09340*.

Shapley, Lloyd S (2016). *17. A value for n-person games*. Princeton University Press.

Street, W Nick, William H Wolberg, and Olvi L Mangasarian (1993). "Nuclear feature extraction for breast tumor diagnosis". In: *Biomedical image processing and biomedical visualization*. Vol. 1905. International Society for Optics and Photonics, pp. 861–870.

To, Kelvin Kai-Wang et al. (2020). "Consistent detection of 2019 novel coronavirus in saliva". In: *Clinical Infectious Diseases* 71.15, pp. 841–843.

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

Uppal, Amit et al. (2020). "Critical Care And Emergency Department Response At The Epicenter Of The COVID-19 Pandemic: New York City's public health system response to COVID-19 included increasing the number of intensive care units, transferring patients between hospitals, and supplementing critical care staff." In: *Health Affairs* 39.8, pp. 1443–1449.

Vaishya, Raju et al. (2020). "Artificial Intelligence (AI) applications for COVID-19 pandemic". In: *Diabetes & Metabolic Syndrome: Clinical Research & Reviews* 14.4, pp. 337–339.

Van Buuren, Stef (2018). *Flexible imputation of missing data*. CRC press.

Verelst, Frederik, Elise Kuylen, and Philippe Beutels (2020). "Indications for healthcare surge capacity in European countries facing an exponential increase in coronavirus disease (COVID-19) cases, March 2020". In: *Eurosurveillance* 25.13, p. 2000323.

Wang, Wenling et al. (2020). "Detection of SARS-CoV-2 in different types of clinical specimens". In: *Jama* 323.18, pp. 1843–1844.

Wölfel, Roman et al. (2020). "Virological assessment of hospitalized patients with COVID-2019". In: *Nature* 581.7809, pp. 465–469.

Wu, Di et al. (2020). "The SARS-CoV-2 outbreak: what we know". In: *International Journal of Infectious Diseases* 94, pp. 44–48.

Wu, Joseph T, Kathy Leung, and Gabriel M Leung (2020). "Nowcasting and forecasting the potential domestic and international spread of the 2019-nCoV outbreak originating in Wuhan, China: a modelling study". In: *The Lancet* 395.10225, pp. 689–697.

Xie, Xingzhi et al. (2020). "Chest CT for typical coronavirus disease 2019 (COVID-19) pneumonia: relationship to negative RT-PCR testing". In: *Radiology* 296.2, E41–E45.

Yan, Le Cun, B Yoshua, and H Geoffrey (2015). "Deep learning". In: *nature* 521.7553, pp. 436–444.

Ye, Zheng et al. (2020). "Chest CT manifestations of new coronavirus disease 2019 (COVID-19): a pictorial review". In: *European radiology* 30.8, pp. 4381–4389.

Zhou, Peng et al. (2020). "A pneumonia outbreak associated with a new coronavirus of probable bat origin". In: *nature* 579.7798, pp. 270–273.

Zhu, Na et al. (2020). "A novel coronavirus from patients with pneumonia in China, 2019". In: *New England journal of medicine*.

Zou, Lirong et al. (2020). "SARS-CoV-2 viral load in upper respiratory specimens of infected patients". In: *New England Journal of Medicine* 382.12, pp. 1177–1179.