
Norwegian University
of Life Sciences

**Master's Thesis 2019 30 ECTS**
Faculty of Science and Technology

# Preliminary evaluation of using machine learning to prioritise cancer patients for proton radiotherapy by predicting dose to organs at risk

Linda Josephine Claesson
Master of Science in Data Science

# Acknowledgements

This thesis marks the completion of my Master of Science in Data Science at the Norwegian University of Life Sciences (NMBU). The work was done through a co-operation between the Department of Medical Physics at the Oslo University Hospital (OUS), the Department of Physics at the University of Oslo (UiO), and the Faculty of Science and Technology (REALTEK) at NMBU.

Firstly, I want to give my sincere appreciation and gratitude to my head supervisor, Professor Cecilia Marie Futsæther. The process of writing this thesis would not have been as gratifying without her endless support and guidance.

Also, thank you to my second supervisor from NMBU, Associate Professor Oliver Tomic, for his counselling and teachings regarding tools and analyses necessary for the thesis.

Further, I want to thank Doctor Einar Dale from OUS and Professor Eirik Malinen from UiO for accessing and compiling the data set. In addition, I want to thank Aurora Rosvoll Grøndahl for transporting the data set from Oslo to NMBU in Ås.

Thank you to everyone present at the BioRadiance meeting at OUS on the 17th December 2018, which contained insightful discussions and invaluable information regarding my thesis.

Next, I am grateful for all of the advice from Runar Helin and Yngve Mardal Moe, who always helped me when I had questions, no matter how small or comprehensive they were.

Finally, I would like to thank my friends and family for their support this spring. A special thank you to Christine, Erik, Espen, Maria, Sofie and Åshild for encouragement and advice during our writing breaks.

---

Linda J. Claesson

Ås, 13th May 2019

# Abstract

The investment in proton radiation therapy raises the question of how cancer patients should be prioritised for this treatment method. A large advantage to proton therapy is that one can minimise the radiation received by organs at risk. Machine learning might be used for predicting the impact of organs for different radiotherapy methods. Thus, machine learning could be a helpful tool in the prioritisation process.

In this thesis, spatial features were extracted from medical images of head and neck cancer patients. These features were used for analysis of photon dose distributions of target volumes and organs at risk. Additionally, correlations between features were investigated. It was confirmed that the distance between radiation target volumes and organs at risk correlated with the dose received by the organs. Further, the analysis contributed to an understanding of which spatial features were expected to affect the dose given to organs at risk.

Within machine learning, both classification and regression algorithms were tested for dose prediction by using the extracted spatial features. In addition, different combinations of algorithms and features were evaluated. The algorithms with the highest accuracy scores were Logistic Regression and Random Forest.

However, the machine learning models either overfitted or underfitted. Thus, other features and machine learning methods should be tested. A possibility for future work is to use deep learning for constructing spatial features and prioritising patients for proton therapy.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **CT** | Computed Tomography |
| **CV** | Cross-Validation |
| **CNN** | Convolutional Neural Network |
| **CSV** | Comma-Separated Values |
| **CTV** | Clinical Target Volume |
| **DSC** | Dice Similarity Coefficient |
| **EDA** | Exploratory Data Analysis |
| **FN** | False Negative |
| **FP** | False Positive |
| **GTV** | Gross Target Volume |
| **HNC** | Head and Neck Cancer |
| **HU** | Hounsfield Unit |
| **IDE** | Integrated Development Environment |
| **IDL** | Interactive Data Language |
| **IQR** | Inner Quartile Range |
| **KNN** | K-Nearest Neighbours |
| **LASSO** | Least Absolute Shrinkage and Selection Operator |
| **MAE** | Mean Absolute Error |
| **MSE** | Mean Squared Error |
| **NTCP** | Normal Tissue Complication Probability |
| **OAR** | Organ At Risk |
| **OvR** | One-versus-Rest |

**PC** Principal Component

**PCA** Principal Component Analysis

**PRE** Precision

**PTV** Planning Target Volume

**Q1** First Quartile

**Q3** Third Quartile

**REC** Recall

**RMSE** Root Mean Squared Error

**SBS** Sequential Backward Selection

**SBFS** Sequential Backward Floating Selection

**SVM** Support Vector Machine

**TN** True Negative

**TP** True Positive

**UINT** Unsigned Integer

**WL** Window Level

**WW** Window Width

# Chapter 1

# Background

The search for the most optimal cancer treatment is highly prioritised when 17 million people get cancer each year worldwide, and 9.6 million of them die [1]. Although we have a considerable amount of knowledge within oncology today [2], more research is required to improve treatment and survival statistics.

A treatment type used for more than a century is radiation therapy (also called radiotherapy) [3]. The technique uses high doses of radiation for targeting cancer tumours [4]. The standard type of radiotherapy today is photon therapy. However, in the last decades, proton therapy has become more prominent in radiotherapy. Several proton therapy facilities have been built worldwide in the 21st century [5]. Norway has recently decided to invest in two proton centres in Bergen and Oslo [6]. These are set to open in 2023.

The aim of building proton centres in Norway is to offer cancer patients a treatment that can cause fewer side effects than standard photon therapy [6]. Today, only a third of Norwegian cancer patients who are preferred for proton therapy are sent overseas for treatment [7]. Consequently, many cancer patients' acquire side effects which can be painful and burdensome still long after treatment [8] [9] [10]. Thus, the proton therapy centres in Norway will make this favoured treatment more readily available for cancer patients in need.

Proton therapy is a more accurate treatment for targeting cancer tumours than photon therapy, because proton beams can target the tumour more accurately [11][12]. As a result, the damage to surrounding organs and healthy tissue is minimised. Therefore, proton therapy could be a more advantageous solution for many cancer patients, especially if the tumour lies close to vital organs.

A disadvantage of proton therapy is, however, that the estimated cost is two to three times higher compared to standard photon therapy [12]. Moreover, any cancer patient can benefit from proton therapy, but some patients have a greater need than

others [7]. Therefore, it will be necessary to prioritise which cancer patients would benefit the most from proton therapy. This task requires knowledge of which patient factors relate to the largest beneficial gain when using the more expensive proton therapy. Some factors, such as age, are known to affect the decision of prioritising a patient to proton therapy [6] [13]. Additionally, the tumour's location relative to other organs at risk for harmful radiation also weighs in on the prioritisation decision [14]. However, more research is required for explaining other contributing factors [15].

Prioritising patients would require that radiologists consider many different factors for each patient [14]. Such a task could potentially be time-consuming, in addition to being subjective. Therefore, an approach for providing rapid and consistent assessment of the patients could be implemented using machine learning.

Machine learning is a branch of artificial intelligence which focuses on building accurate data models to explain trends in a data set where the computer learns from said data set [16]. In the case of proton therapy, a machine learning model could potentially evaluate all aspects of the patient's condition, and thus prioritise the patients most in need of proton therapy. Thereby, the computer could perform many calculations, and predict which patients would benefit from proton therapy. The radiologists would then need to assess the model's results. Nevertheless, the radiologists work load would decrease.

In order to build a machine learning model to predict the optimal form of radiotherapy, it must be trained by using relevant input parameters called *features*. Features (also called variables) are numerical values representing attributes [17]. In the machine learning model example given above, the features would be the attributes of patients, for example age of patient or tumour location. Further, for the machine learning model to be successful, the features must be associated with the model output. In the case of predicting the optimal form of radiotherapy for a patient, one would have to know the traits of a patient indicating the need for proton therapy. Thus, before constructing the machine learning model, features must be extracted from patient data.

Considering using machine learning for the purpose of proton therapy prioritisation is new in the field of radiation oncology [14]. Thus, this thesis will explore what patient factors extracted from CT images and dose plans are significant for predicting the optimal radiotherapy treatment. The goal of these features is to accurately predict dose given to an organ at risk for each radiotherapy treatment method. This is because the doses to organs largely determines the preferred treatment for a patient [14]. Therefore, a machine learning algorithm could use the dose predictions in the prioritisation process.

Through a collaboration between the Norwegian University of Life Sciences, University of Oslo, and Oslo University Hospital, data concerning 225 patients with

recorded head and neck cancer (HNC) was contributed to this thesis. The variables for describing the head and neck region of the patients were selected through guidance from oncology research and radiologists. Further, the same variables extracted were used for an explorative analysis of dose distributions. Additionally, assumptions concerning the data set was also conducted.

# Chapter 2

# Theory

## 2.1 Radiotherapy: protons versus photons

The most common form of radiation therapy thus far is based on photons [18]. The radiotherapy method uses high-energy x-rays (electromagnetic radiation) [19], which yield parcels of energy called photons [20]. Photon therapy has proven quite successful in treating cancer. However, the manner by which photons deposit energy in matter results in irradiation of surrounding healthy tissue and organs, which is harmful to the patient [12]. Thus, the interest for protons arose. The method using protons was first introduced in the 1950's, but unrealistic to perform at the time due to both technological [21] and financial [22] limits. Today, even though the equipment and procedures are still very expensive, researchers are looking into when the costs might be worth it.

The reason as to why using protons is theoretically a better choice is illustrated in Figure 2.1 on the following page. When a patient is irradiated with photons, most of the energy is deposited at the beginning of the beam. As the x-ray penetrates deeper into the body, there is a steady drop of energy deposited. Also, there is an exit dose beyond the tumour [12]. Thus, tissue surrounding the tumour may receive a considerable radiation dose, resulting in tissue damage. On the other hand, the amount of energy deposited when using protons is at its maximum deeper into the patient's tissue [12], at the curve's *Bragg peak*. Further, a tumour can be irradiated by proton beams using different energies, which achieves what is called a *spread-out Bragg peak* [12]. This is done to irradiate the different depth levels of the tumour, resulting in the tumour being irradiated as much as possible, with minimal irradiation of the surrounding tissue.

Nonetheless, proton therapy has a large investment cost to consider. The cost for a proton facility is 3.2 larger than for a photon facility [23]. The construction cost for the two proton therapy centres to be built in Norway is estimated to be 2.9 billion

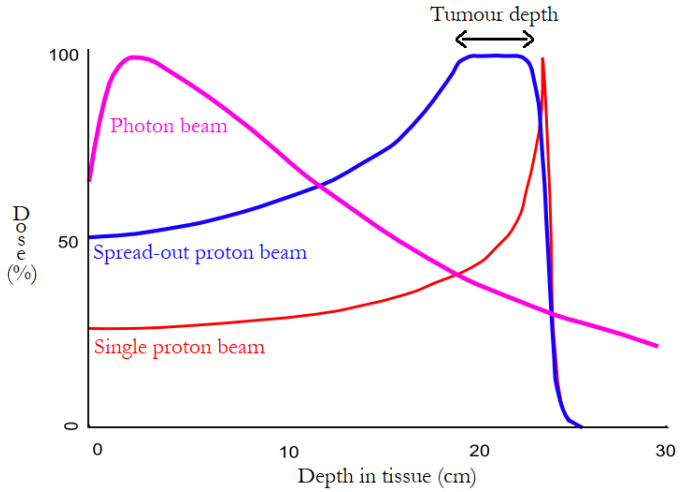



Figure 2.1: Illustration of dose deposited in tissue as a function of depth by a photon beam, single proton beam and spread-out proton beam.

"BraggPeak" by Dr A A Miller \CC BY-SA 3.0. Edit: Other explanations were added to the graphs.

NOK [24]. Nevertheless, since proton therapy can achieve accurate dose delivery, this can result in fewer complications for the patients, and therefore fewer medical treatments later in life [25]. Proton therapy might, therefore, save money in the long run. Although this might be true, there is no randomized trial yet to prove this [26]. Thus, access to proton centres for cancer patients remains limited.

Due to the high costs and limited access, a prioritisation of patients who would benefit the most from proton therapy must be done [27]. Some patients are already prioritised over others, such as children [6]. This is because children are still physically developing, and harming healthy organs and tissue can have a greater negative effect than to irradiate the same organs in an adult [13]. Furthermore, some organs can be irradiated without too many after-effects or complications. There are some organs which should be completely left unirradiated [14]. Section Section 2.3 on page 7 describes the relevant organs at risk to consider for this thesis.

## 2.2 Different target volumes in radiotherapy

When planning a radiotherapy session, the volume which needs to be irradiated is referred to as the *target volume*. Further, the target volume is divided into four subcategories: the gross target volume (GTV), clinical target volume (CTV), internal target volume (ITV) and planning target volume (PTV) [28].

GTV and CTV are biological concepts [29]. The GTV refers to the visible part of the tumour in medical images. The CTV is the suspected extension of the tumour, which may also include surrounding lymph nodes.

The ITV and PTV are volumes which compensate for irregularities and uncertainties related to the differences between the planning dose and actual irradiated area [28]. The CTV may vary in size, shape and position, therefore an internal margin must be added, thus resulting in the ITV. Further, the PTV is defined to select appropriate beam settings. As a result, PTV ensures the desired irradiation dose is given to the tumour. The difference in target volumes is illiustrated in Figure 2.2

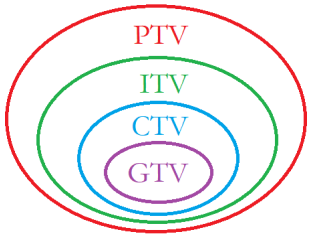



Figure 2.2: Illustration of the different target volume definitions.

## 2.3 Organs at risk for head and neck cancer

Radiotherapy usually affects organs and tissue surrounding or close to the PTV [9][8][10]. These organs are referred to as organs at risk (OAR), where some OARs are given a higher protection priority than others [14]. Which OARs that are prioritised vary depending on the location of the PTV and other patient specific factors. However, since this thesis will only look at HNC, only OARs in this area will be discussed. Further, there might be some OARs in HNC not mentioned in this thesis. Nonetheless, the OARs discussed will be organs that have reported to be focused on in oncology research [9][8][10][14].

An OAR that should have a limited amount of irradiation is the medulla oblongata of the brain stem, shown in Figure 2.3. The reason for this is that the medulla plays a vital role for several basic functions of the body, such as blood flow and breathing [30]. Therefore, no harm should be done to the medulla, and one needs to be strict using small irradiation doses only [14].

More common concerns for radiotherapy treatment after-effects for HNC patients are xerostomia (dry mouth) and sticky saliva. Many patients have trouble with these issues years after treatment [8].

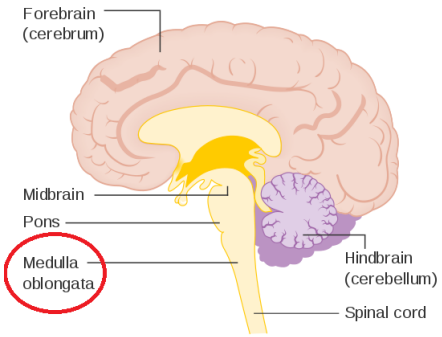



Figure 2.3: Illustration of the location of the medulla oblongata, a part of the brain stem. The name of the medulla is circled in red.

"Diagram showing the brain stem which includes the medulla oblongata, the pons and the midbrain (2) CRUK 294" by Cancer Research UK \CC BY-SA 4.0. Edit: Added red circle.

The OARs found to be related to these issues are the salivary glands, meaning the parotid, submandibular and sublingual glands, as shown in Figure 2.4.

Despite the fact that these after-effects are bothersome for the patient, they are not life or death critical. Also, some patients who end up with xerostomia or sticky saliva improve over time [8]. Therefore, the dose given to the salivary glands should be minimal, but is not as big of a priority as the medulla is. However, if one salivary gland is located such that it receives a large radiation dose, measures should be taken to spare the other one [31].

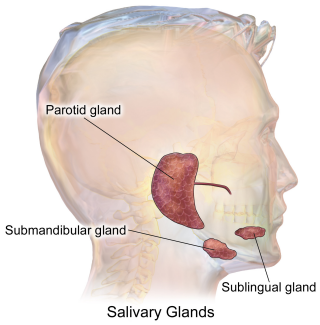
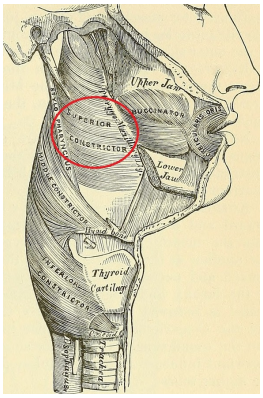



Figure 2.4: Side-view of the three main types of salivary glands. Humans have a pair of each type of salivary gland shown, the left and the right side, giving six main salivary glands in total.

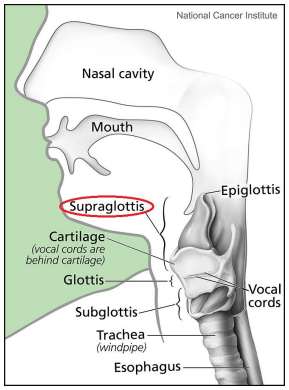
"Blausen 0780 SalivaryGlands" by Bruce Blaus \CC BY 3.0.

HNC patients may also struggle with dysphagia (swallowing difficulty) after radiotherapy [9]. The mean doses to the superior pharyngeal constrictor muscle (Figure 2.5a) and the supraglottic larynx (Figure 2.5b) were found to be the most accurate OAR predictors of whether the patient would get dysphagia [9]. Further, other factors, such as age, also affect the likelihood for the patient to develop dysphagia. Thus, more research is needed on this topic. However, the organs mentioned are still OARs to be considered when doing radiotherapy to minimise the risk of dysphagia.




(a) Side-view of the superior constrictor muscle. The superior constrictor muscle's name and location is circled in red.

"Image from page 398 of *Anatomy, descriptive and surgical* (1887)" by H. Gray, T. Pickering and William Williams \CC BY 2.0. Edit: Added red circle.




(b) Location of the supraglottis in the larynx. The name of the supraglottis is circled in red.

"Larynx and Nearby Structures" by National Cancer Institute \Public Domain. Edit: Added red circle.

Figure 2.5: Illustrations of the locations of the superior constrictor muscle and the supraglottic larynx.

## 2.4 CT images and dose plans

With Computed Tomography (CT) one can achieve image slices of the patient's bones and tissues [32]. CT works by emitting x-ray photons on the patient by having the emitter rotate around the part of the body that needs examining. Directly opposite of the beam is a x-ray detector, receiving the transmitted photons. Thus, by knowing how much of the beam was emitted and transmitted, one can calculate the *attenuation coefficients* of the volumes.

The attenuation coefficient $\mu$ of a material measures how much the beam has weakened (attenuated) when the x-ray photons transmit through the material [33]. The intensity of photons transmitted $I$ is calculated using Equation (2.1), where $I_0$ is the initial intensity of the photons, $\mu$ is the attenuation coefficient, and $x$ is the photons' travelled distance.

$$I = I_0 e^{-\mu x} \tag{2.1}$$

The two-dimensional image slices achieved by CT can be added on top of each other to create a three-dimensional image [32]. A three-dimensional image consists of a grid of voxels (volumetric pixels). Each voxel corresponds to a small volume unit, which depends on the image's resolution. A voxel can contain a value corresponding to the small volume unit. In a CT image, the value of a voxel represents the attenuation coefficient for the specified volume unit [34].

The attenuation coefficient values in a CT image can be used to distinguish type of tissues inside the body [35]. Bones have a high attenuation value, and absorb much of the radiation emitted [36]. Air and water, on the other hand, have small attenuation coefficients, because most emitted photons transmit to the detector.

Before examining CT images, it is common to fit the attenuation values into the Hounsfield scale. This means the values are normalised to the attenuation of water [37], where air has a value of -1000 HU and bone between +700 and +3000 HU [38], where HU stands for Hounsfield Unit.

To enhance the contrast of a specific tissue that is being examined in the CT image, CT windowing is done. How windowing works is well explained by Xue et al. [39]: "Windowing is controlled by two parameters: window level (WL) and window width (WW). (...), only the tissues with HU values within the specified window ([WL-WW/2, WL+WW/2]) are mapped onto the full range of gray scale; the tissues with HU values above (>WL+WW/2) or below window (<WL-WW/2) are set to be all white or all black.". Thereby, windowing enhances the tissues relevant for the examination.

Further, the CT scan of a cancer patient is used to create an individual radiother-

apy treatment plan specific to the patient, called a dose plan [40]. The dose plan illustrates what doses will be given to different volumes of the patient's body. For example, the PTV will be given a high dose value, while the OARs will be given a lower dose. By this, each voxel of a dose plan has an associated dose value, given in Gy, which represents the planned dose to give the voxels.

## 2.5 Machine learning

As stated earlier, machine learning is a data analysis method which can identify patterns, and use these patterns for later predictions [16]. There are three different types of machine learning: supervised, unsupervised and reinforcement learning [41]. In this thesis, we will only focus on supervised learning, and will refer to this machine learning type when discussing machine learning further.

### 2.5.1 Supervised learning

Raschka and Marjaili write in the book *Python Machine Learning* [41] that "The main goal in supervised learning is to learn a model from labelled training data that allows us to make predictions about unseen or future data. Here, the term **supervised** refers to a set of samples where the desired output signals (labels) are already known.". Thus, the data set needs features and labelled output data to train a model. Then, the model can predict labels for unseen data later.

Within supervised learning, there are two subcategories: classification and regression analysis [41]. Within classification, the labels are discrete class label outputs, and are either binary or multiclass labelling tasks. For regression however, outputs are continuous.

A binary classification task is when the machine learning model needs to distinguish between two possible cases [41]. For example, the task could be to decide whether a fruit is an apple or not. Moreover, in binary classification, the class describing the occurrence of an instance is called the positive class, while the class describing a non-occurrence is called the negative class. In the apple example, the instances of apples belong in the positive class, whereas the instances that are not apples belong in the negative class.

When there are more than two possible discrete label outputs, however, it is a multiclass classification task [41]. By extending the apple classification problem to also classify other types of fruit (e.g. pears, bananas, oranges), this would become a multiclass classification problem.

Unlike classification, regression analysis is for predicting continuous responses [41]. Therefore, a regression task involves using features and a continuous response variable to try and find a relationship between them. For example, the hours slept may have a relation to the amount of coffee drunk the next morning. To build a regression model from this hypothesis, the hours slept and coffee consumed would be recorded for a period of time. Then, this data could be used to predict the amount of coffee one is likely to drink, based on the hours of sleep the previous night. Figure 2.6 illustrates a linear regression of the example. However, it is important to note that models in regression analysis can be non-linear as well, meaning the model consists of a non-linear combination between the variables [41].

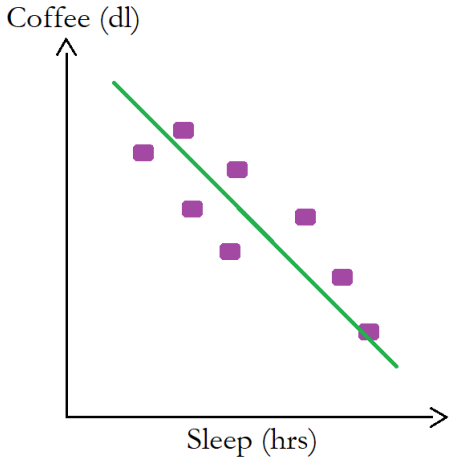



Figure 2.6: Example of a linear regression model, with hours slept as a feature and coffee amount as the continuous response variable. The rounded, purple rectangles represent sample points, and the green line represents a fitted linear regression line of the data.

### 2.5.2 Preprocessing

As Raschka and Marjaili state in *Python Machine Learning* [41]: ”Raw data rarely comes in the form and shape that is necessary for the optimal performance of a learning algorithm.”. A data set needs to be prepared and refined before using it for machine learning. Some preprocessing steps include:

- splitting the data set into training, test and validation sets.
- encoding categorical data values.
- scaling features using either normalisation or standardisation.

In addition, feature extraction, feature selection and dealing with missing values are also elements of preprocessing [41]. However, this thesis will not explain techniques for handling missing values, and only feature selection methods used in the thesis are explained. Nonetheless, these are otherwise important steps in the preprocessing of data.

### Encoding categorical data

There are two subcategories to categorical data: **nominal** and **ordinal** [41]. Ordinal features have values which can be sorted, whereas nominal features cannot be sorted. For example, size and colour can be features to describe t-shirts. The sizes would be an ordinal feature, because sizes are ordered as $S < M < L$. On the other hand, for the nominal colour feature, there is no such order.

For machine learning algorithms to understand categorical values, they need to be described numerically [41]. For ordinal features, one can assign different integers to represent each nominal value. Accordingly, the size feature of the t-shirt could be transformed to $S = 0$, $M = 1$ and $L = 2$. A learning algorithm would interpret the transformed size features as having a particular order, because $0 < 1 < 2$. Therefore, a different approach must be taken with nominal features.

A method for encoding a nominal feature is to create a dummy feature for each unique value the nominal feature has [41]. A dummy feature contains mostly zeros, but are equal to 1 for the instances where the value of the dummy feature is true. The pandas package [42] of Python has a convenient method for creating dummy features called `get_dummies`.

### Splitting the data

In supervised learning, one partitions the data set into training and test set [41]. The training data set is for fitting the machine learning model, and the test set is for prediction using the fitted model. Thus, the model has two performance evaluations, where the test set gives an unbiased evaluation.

Additionally to the training and test set, a part of the training set can be partitioned and used as a validation set [43]. The validation set is used for predicting and estimating the performance of a model for further tuning. The procedure of tuning a model is explained in Section 2.5.6 on page 24.

A visual representation of a data set partitioned into a train, validation and test set is in Figure 2.7 on the next page.

Selecting the sizes of a train, validation and test set is only solved through empirical investigation [44]. However, common split ratios of training and test sets are 60:40, 70:30 and 80:20 (training : test) [45]. When deciding the split ratio, one must avoid allocating too much information to the test set. However, a smaller test set may also lead to a more inaccurate estimation of the fit.

The function `train_test_split` from the scikit-learn package (also known as sklearn) [46] of Python partitions a given data set into separate training and test sets

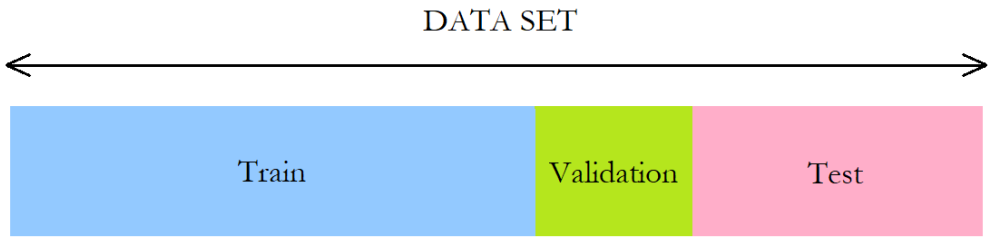



Figure 2.7: A data set partitioned into a train, validation and test set.

[41]. With `train_test_split` one can decide the proportion of the data to be set aside for the test set. Additionally, the parameter `stratify` is used if one wishes that the train and test sets have the same proportions of class labels.

### Feature scaling

Feature scaling is to set the features onto the same scale [41]. If any features have larger scales than the others, then these features will weigh heavier for the optimisation process than the other features. As a result, the features with larger scales have a greater impact on the resulting model. Therefore, the motivation for feature scaling is to ensure that the machine learning algorithm weighs all features equally.

Decision trees and Random Forest (explained in Section 2.5.5 on page 23) are two examples of machine learning algorithms that do not require feature scaling [41]. Nevertheless, most machine learning algorithms do. Therefore, it is an important preprocessing step to consider.

A common approach to feature scaling is **standardisation** [41]. To standardise the features in a matrix $\mathbf{X}$, Equation (2.2) is applied to each value $x^{(i)}$ of each feature column $x$, where $\mu_x$ and $\sigma_x$ are the sample mean and standard deviation of the associated feature column. The result of standardisation is that the scaled feature columns are centred with a mean equal to 0 and a standard deviation of 1 [41].

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x} \tag{2.2}$$

The Python package scikit-learn [46] has an implemented class for executing standardisation called `StandardScaler` [41]. The `StandardScaler` is fitted to the training set, and then used to transform both the training set and test set. The order of fitting and transforming the `StandardScaler` is to ensure that the `StandardScaler` is not influenced by the test set.

### 2.5.3 Overfitting and underfitting

The goal of machine learning is to create a model with a high prediction performance. The cause of a model's poor performance is due to either overfitting or underfitting [47]. Therefore, measures must be taken to prevent overfitting and underfitting.

Overfitting and underfitting is when a model does not generalise well on unseen test data [41]. An overfitted model is unreasonably complex with too many parameters describing it. Thus, overfitted models are sensitive to a data set's randomness (high variance), which leads to poor performance of the test data. On the other hand, an underfitted model is not complex enough to capture the pattern of the training data. There is a systematic error between the true labels and predictions of an underfitted model (high bias). Hence, underfitted models also suffer from poor performance on the test data, additionally to poor performance on the training data. Thus, the goal is to build a robust model that scores almost equally well on training and test data. The differences between overfitted, underfitted and robust regression models are illustrated in Figure 2.8.

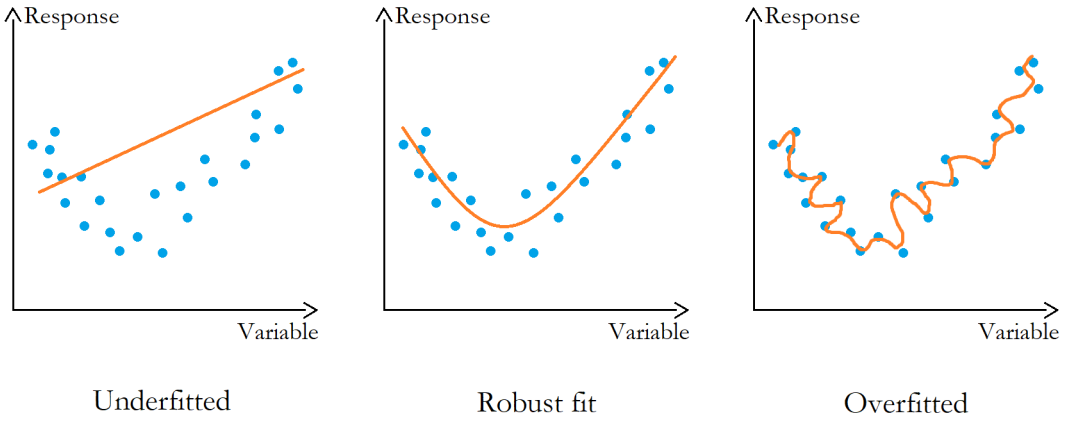



Figure 2.8: Illustration of an underfitted, an overfitted and a robust regression model for a data set. The blue dots represent sample points from a training set, and the orange lines represent different regression models fitted to the data.

There are several methods to prevent overfitting or underfitting a model [41]. For an underfitted model, one tries to increase the complexity, for example by adding model parameters and features describing the data set. Contrarily, one tries to decrease the complexity of an overfitted model. Some techniques include reducing the number of features, ensembling and regularisation. (Examples of what ensembling and regularisation entail are described later in Section 2.5.5 on page 19).

**Techniques for adjusting the dimensionality of a feature subspace**

Polynomial regression is a method for increasing the number of features, thus increasing the complexity of a machine learning model [41]. The method works by combining features as different polynomial terms. The polynomial features are combined up to a certain polynomial degree. For example, if a data set has the two features $[a,\ b]$, the polynomial terms of degree two would be $[1,\ a,\ b,\ a^2,\ ab,\ b^2]$ [46].

Sequential Backward Selection (SBS) is a feature selection algorithm that aims to reduce the number of features [41]. SBS sequentially removes features from the full feature set until the new feature set only has the requested number of features. The algorithm determines which feature to sequentially remove based on which feature combination that results in the lowest difference in performance before and after the removal of a particular feature. In other words, each time SBS needs to remove a feature from the original data set, it removes the feature that causes the least performance loss after removal.

A variation of SBS is Sequential Backward Floating Selection (SBFS) [48]. Similarly to the SBS, SBFS sequentially removes features. However, after the SBFS has found a feature to remove from the feature set, the algorithm looks through the previously disregarded features to evaluate if adding one of the previously removed features will improve the performance. If the performance is improved by adding a previously removed feature, the algorithm adds back said feature to the feature set. The SBFS will then find another feature to remove to ensure that the dimensionality of the feature set is reduced.

### 2.5.4 The cost function and its metrics

To estimate how well a machine learning model is performing, a *cost function* is used. Cost functions measures how well a model is able to predict a class or response by using the associated features [49]. Although cost function types differ, they typically express the performance score as a difference or distance between the predicted and true value. Therefore, the goal of machine learning is to minimise the cost function [41].

Cost functions use different metrics (or scorers) for estimating performance [41]. Moreover, metrics vary for different machine learning problems. Thus, regression problems use different metrics than classification problems.

Additionally, subcategories of a classification or regression problem also impact the decision of a metric [41]. For example, the metric of a binary classification problem might not work for a multiclass classification problem. Further, the decision of

a metric depends on prioritisation of the outcome. This is the case when one for example wants the model to have a higher accuracy score for predicting one class than another.

Due to the large number of metrics, this thesis only explains a few methods of estimating a model's performance.

### Classification metrics

The performance of a classification model can be illustrated using a confusion matrix [41]. The confusion matrix of a binary classification problem is a square matrix with the values for the number of true positives ($TP$s), true negatives ($TN$s), false positives ($FP$s) and false negatives ($FN$s). $TP$ and $TN$ are the number of correctly classified positive and negative class instances, and the $FN$ and $FP$ are the number of incorrectly classified positive and negative class instances. Thus, the number of correctly classified instances are on the diagonal of the confusion matrix. Figure 2.9 illustrates the set-up of a confusion matrix for a binary classification problem.

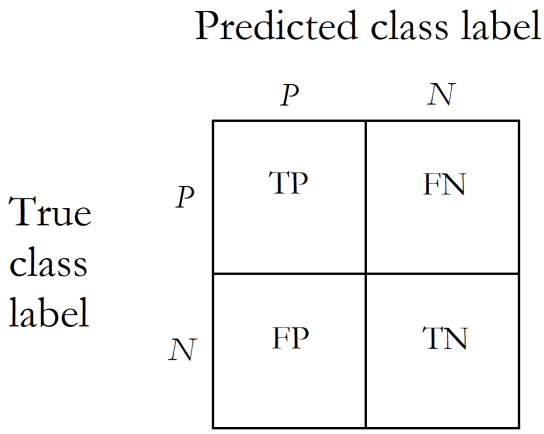



Figure 2.9: Set-up of a confusion matrix for a binary classification problem. $P$ and $N$ represent the positive class label and negative class label. $TP$ = true positives, $FN$ = false negatives, $FP$ = false positives, $TN$ = true negatives.

There are various performance metrics to calculate using the information found in a confusion matrix. Classification accuracy (*ACC*), precision (*PRE*), recall (*REC*) and the *F1* score are four possible measurers of performance [41].

*ACC* is a metric for providing general information about the number of samples that are misclassified [41]. The calculation of *ACC* is the sum of true predictions divided by the number of total predictions. Therefore, *ACC* is defined by Equation (2.3) on the next page, where $TP$ and $TN$ are the true positives and negatives, and $FP$ and $FN$ are the false positives and negatives.

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} \tag{2.3}$$

*PRE* is the fraction of correctly classified positive class instances given the number of all instances that were classified as positive. The calculation of *PRE* is done using Equation (2.4), where $TP$ is the number of true positives and $FP$ is the number of false positives.

$$PRE = \frac{TP}{TP + FP} \tag{2.4}$$

*REC*, on the other hand, is the fraction of correctly classified positive class instances given the number of all positive class instances there are in the data set [41]. Equation (2.5) is used for the calculation of *REC*, where $FN$ is the number of false negatives.

$$REC = \frac{TP}{FN + TP} \tag{2.5}$$

The *F1* score, which is a combination of *PRE* and *REC*, is the precision metric often used in practice, and is defined by Equation (2.6) [41]. The values of an *F1* score are between 0 and 1, where 1 is the highest *F1* score a model can achieve.

$$F1 = 2\frac{PRE \times REC}{PRE + REC} \tag{2.6}$$

The advantage of the *F1* score over *ACC* is that the *F1* score is a balance between *PRE* and *REC* [50]. *ACC* does not represent the misclassifications. In many classification cases, it is also important to detect misclassified instances, and not only measure the model's accuracy. Therefore, a score which uses all $TP$, $TN$, $FP$, and $FN$ would be a solution to the problem of *ACC*.

Although the metrics explained above are specific to binary classification problems, it is possible to extend them to multiclass classification problems [41]. One-versus-Rest (OvR) is a technique that extends a binary classifier to multiclass classification problems. This is done by training a single classifier per class, where each specific class is treated as the positive class and the rest as negative classes. Therefore, by averaging the results after an algorithm has used OvR, one can calculate the *F1* score of a multiclass problem.

There are two different averaging methods to choose from when calculating the *F1* score for a multiclass classification problem [41]. The *micro average* is calculated

using the $TP$s, $TN$s, $FP$s and $FN$s of each trained classifier from the OvR. Contrarily, the *macro average* is calculated by adding the performance score from each classifier and dividing by the number of classifiers used. If one wants to weigh each class label equally, the micro average is preferred.

Also, after using the OvR technique, a confusion matrix for a multiclass classification problem can be constructed [46]. Similarly to the confusion matrix of a binary classification problem, the dimensions reflect the number of classes a classification problem has. Thus, a confusion matrix for a multiclass classification problem is a square matrix and at least $3 \times 3$. Therefore, if a multiclass classification problem has five classes, its confusion matrix would be $5 \times 5$, and the number of correctly predicted instances would appear on the diagonal of the confusion matrix.

### Regression metrics

The errors of a regression model are called *residuals* [41]. A residual of an instance $i$ is defined as $e^{(i)} = y^{(i)} - \hat{y}^{(i)}$, where $y^{(i)}$ and $\hat{y}^{(i)}$ are the true and predicted response values of the instance $i$, respectively.

A regression model's performance can be measured by estimating its Mean Square Error ($MSE$) [41]. The $MSE$ is calculated by Equation (2.7), where $e^{(i)}$ is the residual of instance $i$, and $n$ is the number of instances in the data set. The $MSE$ value of a regression model should be as low as possible, which would indicate a high model performance. If the $MSE$ of a model is equal to 0, then the model fits the data perfectly.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(e^{(i)})^2 \tag{2.7}$$

Although $MSE$ is a simple metric, the squaring of the errors can result in a large $MSE$ value due to a single large residual [51]. The root mean squared error ($RMSE$) is a variation of the $MSE$ that is less sensitive to deviant residuals. $RMSE$ is defined in Equation (2.8), where $e^{(i)}$ is the residual of instance $i$, and $n$ is the number of instances in the data set. Similarly to $MSE$, a low $RMSE$ value signifies an accurate regression model, and the lowest $RMSE$ value a model can have is 0.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(e^{(i)})^2} \tag{2.8}$$

Both the $MSE$ and $RMSE$ have the disadvantage that they are dependent on the scale of the data set [51]. For example, a $MSE$ value of 27 can be large for one data

set, while it is a low *MSE* value for another data set. As a result, it can be difficult to know how accurate a model is. Therefore, the coefficient of determination, often referred to as the $R^2$ score, is a metric that is independent of a data set's scale. Thus, the $R^2$ score makes it easier to interpret a model's performance.

The $R^2$ score can be understood as a standardised version of the *MSE*, and is calculated using Equation (2.9), where $Var(y)$ is the variance of the response $y$ of the data set [41]. When *MSE* of a model is 0, then the $R^2$ is equal to 1, meaning if a model fits the data perfectly, then the $R^2$ score is equal to 1. The $R^2$ is bounded between 0 and 1 for the training data set, but can be negative for the test set. If the $R^2$ is negative, it means that the $MSE > Var(y)$, and thus the errors of the test set are larger than the variance of the response values.

$$R^2 = 1 - \frac{MSE}{Var(y)} \tag{2.9}$$

Finally, the Mean Absolute Error (*MAE*) is a performance metric which can be a valid choice over *MSE* if there are outliers in the data set [51]. *MAE* is defined by Equation (2.10), where $e^{(i)}$ is the residual of instance $i$, and $n$ is the number of instances in the data set.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |e^{(i)}| \tag{2.10}$$

### 2.5.5 Algorithms

A machine learning algorithm is a method to use for executing learning of data [52]. There are many learning algorithms to choose from, therefore, only a few of them are explained in this thesis.

#### Logistic Regression

An explanation of how a logistic regression algorithm learns is easier to understand knowing how the simpler perceptron algorithm works.

In the example of a binary classification for a perceptron, there are two possible outcomes, 1 (positive class) and $-1$ (negative class) [41]. To decide between 1 and $-1$, a decision function $\phi$ is used. A decision function takes in data set values of an instance $\mathbf{x}$, and decides which class $\mathbf{x}$ belongs. $\mathbf{x}$ is defined as

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

where $m$ is the number of features of an instance $\mathbf{x}$.

However, a decision function $\phi$ must also determine how much each feature should be weighted to make a correct classification choice [53]. Therefore, the input $z$ of a decision function is written as a linear combination of the values $\mathbf{x}$ and their weights $\mathbf{w}$. Hence, the input $z$ is equal to $w_1x_1 + w_2x_2 + ... + w_m,x_m$, where $m$ is the number of features, and the weights vector $\mathbf{w}$ is defined as

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

A decision function also requires a threshold value [41]. If an input $z$ of the decision function $\phi(z)$ is greater than a certain value $\theta$, then $\phi(z)$ should be equal to 1. Alternatively, $\phi(z)$ should be equal to $-1$. Thus, the output equation of $\phi(z)$ is given by Equation (2.11).

$$\phi(z) = \begin{cases} 1 & if\ z \geq \theta \\ -1 & otherwise \end{cases} \tag{2.11}$$

A simpler version of Equation (2.11) is acquired by redefining $z$ [41]. By adjusting Equation (2.12) into Equation (2.13), and then redefining $\theta$ as explained in Equation (2.14), then Equation (2.11) can be defined as Equation (2.15).

$$z = w_1x_1 + w_2x_2 + ... + w_m,x_m \geq \theta \tag{2.12}$$
$$z = -\theta + w_1x_1 + w_2x_2 + ... + w_m,x_m \geq 0 \tag{2.13}$$
$$z = w_0x_0 + w_1x_1 + w_2x_2 + ... + w_m,x_m \geq 0 \quad w_0 = -\theta, x_0 = 1 \tag{2.14}$$

$$\phi(z) = \begin{cases} 1 & if\ z \geq 0 \\ -1 & otherwise \end{cases} \tag{2.15}$$

Thus, the weights **w** for the linear combination of input value $z$ are important for correctly classifying the instance **x** [41]. Essentially how a perceptron learns is by comparing the true class labels with the predicted class labels. Consequently, the perceptron will adjust the weights in order to make more accurate predictions with future data.

The learning approach is the difference between a perceptron and the logistic regression algorithm [41]. Despite its name, logistic regression is a classification algorithm. Like many other learning algorithms, logistic regression relies on another function to update the weights. The *activation function* of logistic regression is called a logistic sigmoid function (abbreviated as the sigmoid function). The sigmoid function is defined in Equation (2.16), where $z = \sum_{i=0}^{m} w_i x_i$. Plotting Equation (2.16) for $z = [-6,6]$ results in Figure 2.10.

$$\phi(z) = \frac{1}{1+e^{-z}} \tag{2.16}$$

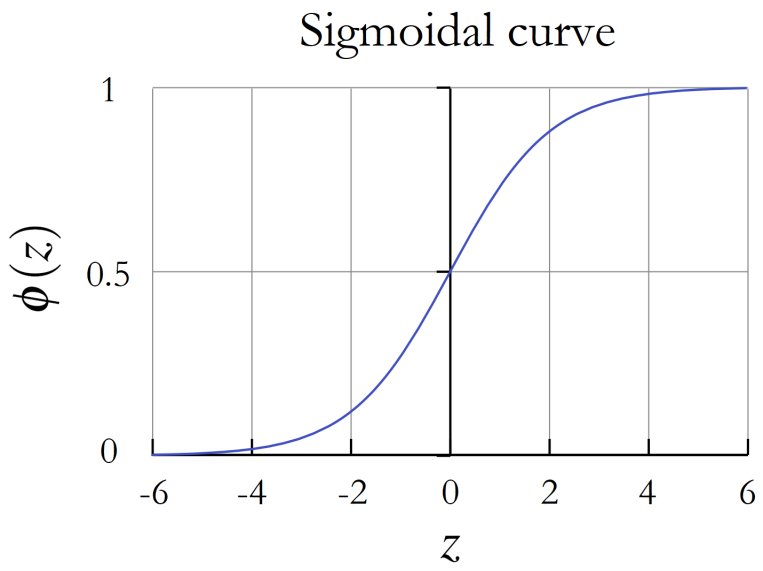



Figure 2.10: Plot of a sigmoid curve. $\phi(z)$ from Equation (2.16) for the input values $z = [-6,6]$ results in output values between $\phi(z) = [0,1]$.

"Logistic-curve" by Qef (Wikimedia Commons user) \Public domain. Edit: Changed axes values, and added title and axes labels.

As illustrated in Figure 2.10, the sigmoid function takes in real values, and transforms them into values in the range of 0 and 1, with 0.5 as the intercept. The output of $\phi(z)$ can be interpreted as the probability that a sample $z$ belongs to class 1 [41].

While the sigmoid function is used as an activation function, logistic regression still uses a threshold function for prediction [41]. The threshold function, Equation (2.17) on the following page, converts the output of the activation function into a binary outcome. $\hat{y}$ is the predicted outcome and $\phi(z)$ is the output of the sigmoid

function. Equivalently, the threshold function to a logistic regression can be written as Equation (2.18), where $\hat{y}$ is the predicted outcome, and $z$ is the linear combination of a data sample and its weights. Thus, for a logistic regression, the positive class is 1, and the other class is 0.

$$\hat{y} = \begin{cases} 1 & if\ \phi(z) \geq 0.5 \\ 0 & otherwise \end{cases} \tag{2.17}$$

$$\hat{y} = \begin{cases} 1 & if\ z \geq 0.0 \\ 0 & otherwise \end{cases} \tag{2.18}$$

Logistic regression uses regularisation to tune the complexity of the model [41]. The idea behind regularisation is to add additional information (bias) to the cost function for penalising extreme weight values. The common L2 regularisation adds the term $\frac{\lambda}{2}||\mathbf{w}||^2$ to the cost function, where $\mathbf{w}$ are the weights in vector form, and $\lambda$ is the regularisation parameter. The regularisation parameter $\lambda$ controls the regularisation strength.

When using logistic regression, the parameter $C$ can be tweaked, which is defined as the inverse of $\lambda$ [41]. Thus, decreasing the value of $C$ leads to the increase in the value of $\lambda$ and also the regularisation strength.

Lastly, the logistic regression classifier can be used for a multiclass classification problem by using the OvR technique introduced in Section 2.5.4 [41].

### Random Forest

The Random Forest algorithm is a machine learning technique which combines multiple decision trees [41]. A decision tree splits the data on a value of a feature to differentiate the samples from each other. This procedure is done repeatedly for various values of distinctive features until the data samples are distinguishable. Consequently, the result of a decision tree is a tree-like structure where the final nodes are the predicted outcome of a sample. An example of a decision tree created for the prediction of Titanic survivors based on passenger information is illustrated in Figure 2.11.

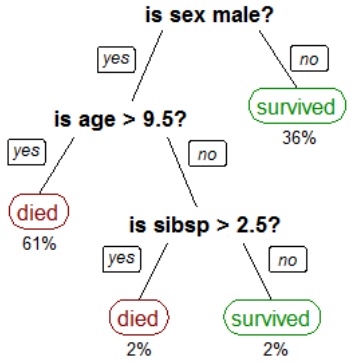



Figure 2.11: Example of a decision tree for prediction of Titanic survivors using the passengers' data. The bold statements are evaluation statements regarding features for splitting the data. The coloured nodes are predicted outcomes. Below each node is the associated percentage (rounded up) of the samples categorised into that specific node.

"CART tree titanic survivors" by Stephen Milborrow \CC BY-SA 3.0. Edit: Added 'yes' and 'no' to all the branches, and only kept percentage below the nodes.

Moreover, the Decision Tree algorithm splits the data using the most descriptive features, thus achieving informative splits [41]. However, the number of splits and the depth of a tree also impacts the usefulness of the splits. As a result, decision trees may easily lead to overfitting.

A solution option for preventing overfitting when using Decision Trees is to use an ensemble of decision trees for prediction, namely a Random Forest algorithm [41]. The idea behind ensembling is to combine classifiers into a meta-classifier, resulting in a better generalisation performance than each individual classifier alone. Therefore, the decision trees that individually suffer from high variance will generalise better when assembled together.

The "randomness" in a Random Forest is from the procedure of building the decision trees [54]. A random sample of $m$ predictors are chosen from the full set of $p$ features. For example, a typical set of $m$ are $m = \sqrt{p}$. A new sample $m$ is chosen for each split. Thus, by forcing the algorithm to evaluate different features for each split, decision trees become less correlated. Consequently, ensembling the final decision trees reduces variance, and hence the Random Forest becomes more reliable.

The prediction of an unseen sample is predicted by the majority vote of the decision trees used in the Random Forest [41]. Thus, Random Forests have lower interpretability than the Decision Tree algorithm, yet gains low susceptibility to

overfitting.

Additionally, there is only one important parameter that requires tuning in a Random Forest algorithm, which is the number of trees to use for the Random Forest [41]. Thus, it is more difficult to create an overfitted Random Forest.

Nevertheless, it is also possible to tune other parameters than number of trees for a Random Forest [41]. The algorithm uses a function to measure the quality of a split, and which function to use can be selected by the data scientist. Plus, the equation for deciding number of $m$ predictors of the total set $p$ to use for potential tree splits can also be decided.

Although the example given in Figure 2.11 for a decision tree was for a binary classification task, Decision Trees and Random Forests can also be used for regression [41]. One difference between the Random Forest classifier and regressor is the types of functions one can use for measuring the quality of a split. For regression these are *MAE* and *MSE* [46]. Also, the predicted target response is calculated using the average prediction of all decision trees in Random Forest regression.

### 2.5.6 Hyperparameter tuning

As aforementioned, a learning algorithm has different parameters that are specified by the user. For example, the number of decision trees used in a Random Forest, or the parameter $C$ for Logistic Regression, are parameters that must be chosen. These type of parameters are called hyperparameters [41].

Varying the values of hyperparameters makes the algorithms behave differently. For example, the hyperparameter $C$ decides the complexity of a Logistic Regression. Moreover, different hyperparameter combinations are optimal for distinct data sets. Therefore, tuning the algorithms' hyperparameters to be optimal for a specific data set helps with the resulting prediction performance [41].

Hyperparameter tuning can be done using a grid search [41]. A grid search is a search paradigm which evaluates model performance for all combinations of the training set with specified hyperparameter values. From this, the grid search obtains the optimal combination of hyperparameter values possible for the training set.

However, the hyperparameter values combination might be optimal only for the given training set, which leads to overfitting [55]. Therefore, an approach for avoiding overfitting is to use cross validation (CV) in the grid search.

The most common form of CV is K-Fold CV [55]. A K-Fold CV splits a training set into $K$ number of subsets, which are referred to as folds. The model used in the CV is iteratively fit $K$ number of times. Each time the model is fitted, a unique

combination of $K-1$ number of folds are used for training. The withheld fold is used for evaluating the fit (a validation fold) with the use of a specified performance metric. A visual representation of the folds for each fit for a 5 fold CV is shown in Figure 2.12.


Figure 2.12: A training set divided into folds for a 5 fold CV. The blue box illustrates a training set, the purple a training fold, and beige a validation fold.

## 2.6 Features of objects in images

There are many features one can extract from image data. For example, edges and point patterns in an image give valuable information for an image recognition problem. However, as discussed, the features need to be related to the goal set. Therefore, the features explained here are the ones relevant for the purpose of this thesis, which are size and distance measurements.

### 2.6.1 Volume

The most straightforward method of calculating the size of an object in an image is by simple addition. A three-dimensional image consists of a grid which is made up of voxels. An object delineated in the image would therefore consist of a certain number of voxels. For example, an object can be defined by 50 voxels in a three-dimensional image. As each voxel corresponds to a small volume unit, for example 1 $mm^3$, the 50 voxels would equal 50 $mm^3$. Thus, adding up the number of delineated voxels will result in the volume of the object.

**Overlapping volumes**

The volumes of delineated objects in an image may overlap. A simple way to numerically quantify the overlapping volume is to calculate the intersection of the objects' volumes. Thus, the overlapping volume is defined as $|A \cap B|$, where $A$ and $B$ are two volumes composed of voxels.

Another technique to describe the overlapping volume is by using the Dice Similarity Coefficient, DSC. DSC is defined by Equation (2.19), where $A$ and $B$ are two volumes composed of voxels [56]. DSC in the medical physics world today is mostly used to evaluate how well an automatic segmentation method completes its tasks [56] [57]. However, DSC is simply a measure of similarity between two samples. As such, Equation (2.19) can be used as a measure of overlapping voxels of two objects. If $Dice(A,B)$ from Equation (2.19) is equal to 1, the voxel sets of $A$ and $B$ are identical, whereas if $Dice(A,B) = 0$, $A$ and $B$ have no voxels in common.

$$Dice(A,B) = \frac{2|A \cap B|}{|A| + |B|} \tag{2.19}$$

### 2.6.2 Euclidean distance and its variations

The distance between the tumour and OARs will affect the radiation dose received by the OARs [14]. This distance can be calculated using several different techniques. However, the goal is to limit the number of features for the machine learning model. Therefore, it is important to use a distance measurement technique that results in the strongest correlation between dose given to the OAR and distance to the PTV.

The Euclidean distance method, which is based on Pythagoras theorem, is one of the most used distance calculation methods [58]. This is because the technique is easy to understand and simple to use for any dimension. In a two-dimensional space one would use Equation (2.20) to calculate distance $d_{ij}$ between the points $(x_i, y_i, z_i)$ and $(x_j, y_j, z_j)$ in the $xyz$- space.

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{2.20}$$

Even though the Euclidean distance method is simple to use for two points, it becomes slightly more complicated for calculating the distance between two objects or regions. This is because one would also need to choose which points within the regions to calculate the distance between. The first choice might be the centre points. However, it is also possible to calculate the minimum or maximum distance

between the regions, by choosing the points that result in the smallest or largest distance possible. The centre-to-centre distance and minimum centre-to-centre distance is illustrated in Figure 2.13.

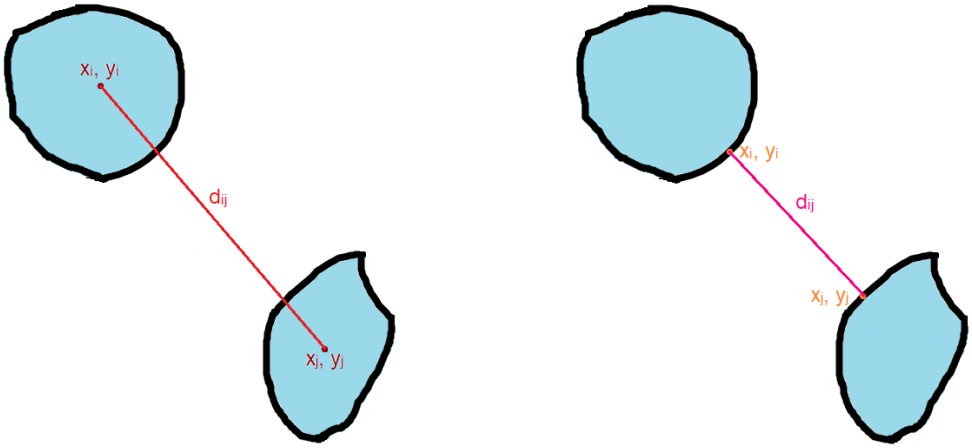



Figure 2.13: Two regions in the $xy$- plane with points to measure the distance between. The points $(x_i, y_i)$ and $(x_j, y_j)$ are chosen to be either (a) points in the centre of the regions, or (b) points in the regions that result in the smallest possible distance between them.

The choice of distance depends on which technique would give the most relevant information. However, a disadvantage for calculating the minimum or maximum distance is that this requires more computational resources. When calculating the distance between the centre points, one needs to first find the centres of the objects and then use Equation (2.20). On the other hand, to calculate the minimum or maximum distance, several distances need to be calculated before determining which points of the object results in the minimum/maximum distance.

## 2.7 Exploratory Data Analysis methods

Exploratory Data Analysis (EDA) is the approach of exploring a data set using methods and tools which illustrate the features and values, thus giving insight into the data at hand [41]. It is recommended to do at least a few EDA techniques before training machine learning models. Considering that there are many EDA techniques available, the methods discussed here will simply be the ones chosen to perform on the HNC data set. The only EDA tools used on the HNC data set that are not explained below are scatter and histogram plots.

### 2.7.1 Box plot

A box plot illustrates the range of values for a data set by using its minimum, maximum, median, first quartile (Q1) and third quartile (Q3) value [59]. Q1 and Q3 are also known as the 25th and 75th percentile. As shown in Figure 2.14, the central rectangle of a box plot spans Q1 to Q3, and is referred to as the inner quartile range (IQR). Also, the median value's location in the IQR is demonstrated. Further, the span between the minimum value and Q1, and the span between Q3 and maximum value are called *whiskers*. The reach of the whiskers are usually defined as $Q1 - 1.5 \cdot IQR$ and $Q3 + 1.5 \cdot IQR$. This means that values smaller than $Q1 - 1.5 \cdot IQR$ or greater than $Q3 + 1.5 \cdot IQR$ are regarded as outlier samples, which can be plotted as individual points.

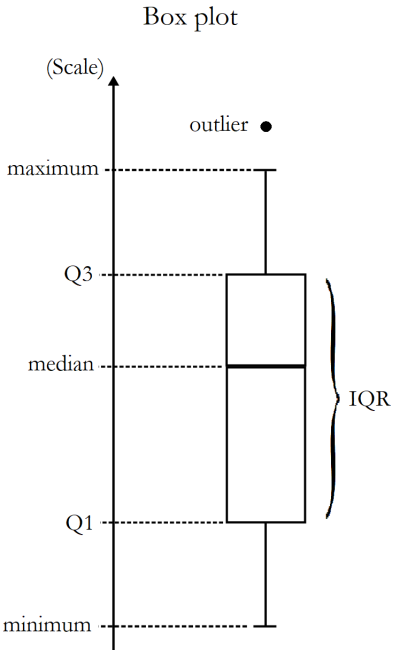



Figure 2.14: Model of a box plot.

### 2.7.2 Correlation methods

The most widely used method of calculating the correlation between a pair of features is by using the Pearson product-moment correlation formula [60]. The Pearson correlation coefficient $r_p$ gives a measurement of the linear correlation between the pair of features. $r_p$ is determined using Equation (2.21), where $\sigma_{xy}$ is the covariance between the features $x$ and $y$, and $\sigma_x$ and $\sigma_y$ are the features' standard deviations. Thus, $r_p$ is in the range of -1 and 1, where $r_p = 1$ is a perfect positive linear correlation, and $r_p = -1$ is a perfect negative linear correlation.

$$r_p = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \tag{2.21}$$

If there are more than two features between which to calculate the pair-wise correlation, a *correlation matrix* may be used. A correlation matrix is a symmetric matrix containing all of the correlation coefficients of each pair of features [41]. Thus, a correlation matrix gives a simple overview of how all the features available are correlated.

Another similar correlation method to the Pearson coefficient is the Spearman rank correlation [61]. The Spearman rank $r_s$ measures the monotonic relationship between

a pair of features. A relationship is monotonic when the function describing the relationship is entirely non-increasing or non-decreasing [62] (in other words, the function's first derivative does not change sign). Hence, a linear relationship is monotonic. Thus, both the Pearson and Spearman methods are similar, because they are both measurements of monotonic relationships. Unlike the Pearson method, the pair of features do not need to be linearly correlated when using Spearman's correlation method. Similarly to the Pearson coefficient, $r_s$ have values between $-1$ and 1, where $|r_s| = 1$ suggests a strong monotonic correlation.

### 2.7.3 Principal Component Analysis

Principal Component Analysis (PCA) is an exploratory and visualization technique for emphasising the variation of a data set [63]. Usually a coordinate system used to illustrate a data set is based on the variables the data has. An example of this is shown in Figure 2.6 on page 11, where the axes explain coffee consumption and hours slept. On the other hand, PCA creates a new coordinate system, with orthogonal axes that correspond to the maximum variation of the data [64]. The axes of the new coordinate system are called principal components (PCs). Thus, PCs are individual combinations of the original variables.

As an example, if three PCs were to be created from a data set, the procedure for creating the PCs would be as follows:

1. The first PC constructed corresponds to the direction of the largest variation of the data set.

2. The second PC corresponds to the direction of the largest variation, given the constraint that it is orthogonal to the first PC.

3. The third PC corresponds to the direction of the largest variation, given the constraint that it is orthogonal to the first and second PC.

The pattern continues as the number of PCs to construct increases.

PCA is useful for feature extraction and reduction of the original feature space [41][64]. Each PC contributes to a certain percentage of the total variance of a data set. The first PCs account for the largest share of the variability. If the few first PCs can describe much of the variability, then the dimensionality of the original feature space can be greatly reduced by using the chosen few PCs as variables [64].

### Described by matrices

A PCA is a singular value decomposition of a data set [65]. A matrix $\mathbf{X}$ can contain data where each column represents a variable, and each row represents an observation. The matrix $\mathbf{X}$ can then be written as $\mathbf{X} = \mathbf{TP^T} + \mathbf{E}$, where the samples of the new coordinate system is represented in matrix $\mathbf{T}$, the new variables in matrix $\mathbf{P}$, and the unexplained residuals in matrix $\mathbf{E}$. Accordingly, the orthogonal set of values in $\mathbf{T}$ describes the relationships between the samples, and the orthonormal set of values in $\mathbf{P}$ describe the relationships between variables [66]. The values in $\mathbf{T}$ and $\mathbf{P}$ are called scores and loadings, respectively.

### Explained variance

As aforementioned, each PC contributes to a certain percentage of the total variance in $\mathbf{X}$. This share of total variance is referred to as explained variance [65]. Therefore, an *explained variance plot* demonstrates the cumulative explained variance achieved by adding the explained variance from each PCs. Thus, the horizontal axis in the explained variance plot is the PC number and the vertical axis represents the variance percentage of $\mathbf{X}$. For example, with the explained variance plot shown in Figure 2.15, the calibrated explained variance for the first two PCs is 70% and 17%. Therefore, the explained variance plot starts at 70 % for PC1, and then increase to 87% for PC2.

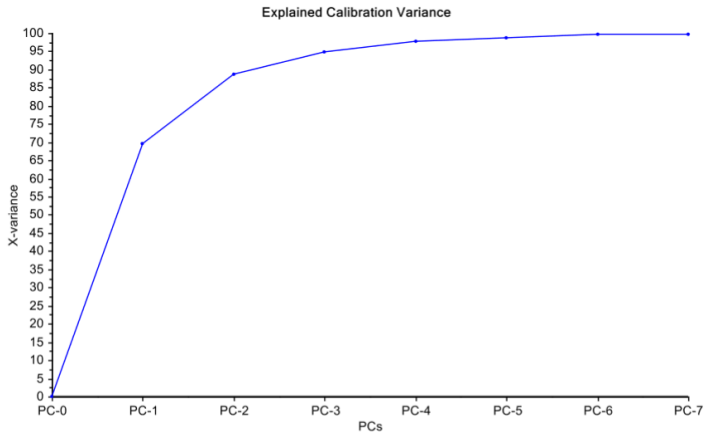



Figure 2.15: Example of an explained variance plot. The blue line illustrates the cumulative explained variance increasing for each PC.

"Explained Calibration Variance" by Oliver Tomic [65] \Permission from owner given via e-mail 06.05.2019.

The curve in an explained variance plot will always reach 100 % if enough PCs are constructed for the PCA [65]. However, if too many PCs are constructed, the PCA may start to try and explain meaningless information that should be classified as noise. Therefore, it could be wise to also plot a validated explained variance

graph in the explained variance plot. The additional curve is constructed by cross-validating the data set when calculating the explained variance for each PC. Thus, the validated explained variance curve can be used for choosing the number of PCs which results in the largest cumulative explained variance, without overfitting the PCA.

### Scores and loadings plots

Before explaining the purpose of the scores and loadings plots, the description of the axes is important for understanding how the plots work. Commonly, the plots are two-dimensional, with distinct PCs as axes [65]. From this, one can interchangeably use different PCs for plotting, and the different plots will demonstrate the distinctive PCs roles in the PCA. For example, a plot using PC1 and PC2 would demonstrate different aspects of the PCA than a plot using PC3 and PC4. However, it is recommended to start plotting with PC1 and PC2 as axes, because these PCs explain the most of the data's variation [66].

Firstly, a *scores plot* is a scatter plot of the columns in matrix **T** [65]. Scores with a short distance from each other in the plot have similar properties, and dissimilar scores are further apart. Additionally, score values in the scores plot relate to the degree a PC explains a score [67].

Plotting the columns of the loadings matrix **P** results in a *loadings plot* [65]. In a loadings plot, variables with a short distance from each other are highly correlated, and variables on opposite sides of an axis are negatively correlated.

A loadings plot also demonstrates the influence each original variable has on the PCs [67]. However, for a more direct illustration of this, a *correlation loadings plot* is created [65]. A correlation loadings matrix is constructed by calculating the correlation coefficients between the elements in matrices **T** and **X**. The result when plotting the correlation loadings matrix is a plot where the correlations between each variable and PC are plotted as values, as illustrated in Figure 2.16 on the following page. Also, because they are correlation coefficients, the values are between $-1$ and 1. The closer a correlation loading is to the origin on a PC, the less influence this variable has on the given PC. In contrast, correlation loadings further away from the origin have a greater influence on the PC.

Additionally, for the correlation loadings plot to be more explicit, circles that correspond to a degree of explained variance are drawn in the plot [65]. Typically, an outer circle will represent 100% explained variance, and an inner circle 50%, as shown in Figure 2.16. Accordingly, an example of a correlation loading on the outer circle means the variable is explained 100% by the PCs.

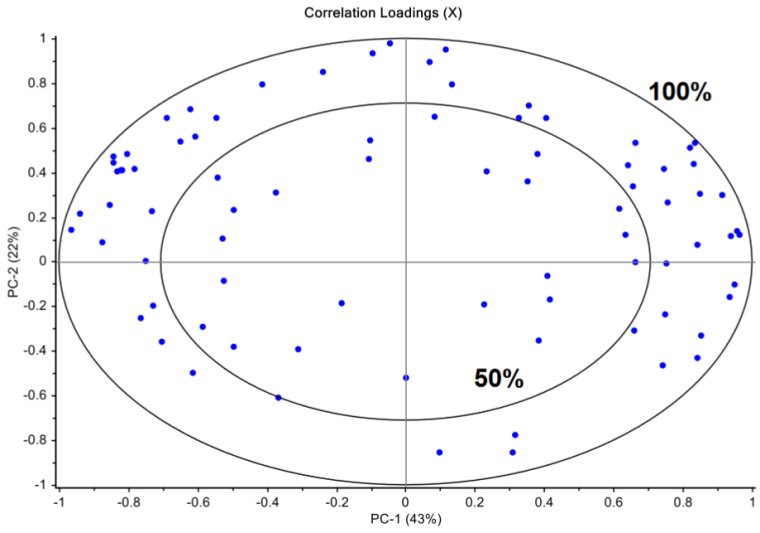



Figure 2.16: Example of a correlation loadings plot. The correlation coefficients between the elements in matrices **T** and **X** are plotted in a coordinate system with the first two PCs as axes. The circles represent the percentage of explained variance, either 50 % or 100 %.

"Correlation Loadings Plot" by Oliver Tomic [65] \Permission from owner given via e-mail 06.05.2019.

Lastly, a *biplot* is a combined scores and loadings plot [68]. Subsequently, because a biplot displays the scores and loadings simultaneously, the interaction between the scores and loadings becomes more apparent when plotted in the same plot.

### Centring and standardisation

The purpose of a PCA is to construct a new coordinate system based on the directions of variance [64]. Therefore, if any variables in $\mathbf{X}$ have a larger variance than others, then these variables will dominate the PCs and plots [65]. To ensure all variables' variances are treated equally, the columns of $\mathbf{X}$ are standardised before the PCA. A value $x_{n,k}$ in $\mathbf{X}$, where $n$ is the row number and $k$ is the column number, is standardised by using Equation (2.22). $\bar{x}_k$ is the mean of the variable in column number $k$, and $\sigma_k$ is the standard deviation of the same variable.

$$Standardised(x_{n,k}) = \frac{x_{n,k} - \bar{x}_k}{\sigma_k} \tag{2.22}$$

However, if the variables in $\mathbf{X}$ use the same scale, then standardisation is not required [65]. Nonetheless, the values of $\mathbf{X}$ are centred for convenience. Equation (2.23) describes how to centre a value $x_{n,k}$, where $n$ is the row number, $k$ is the column number, and $\bar{x}_k$ is the mean of the variable in column number $k$.

$$Centred(x_{n,k}) = x_{n,k} - \bar{x}_k \tag{2.23}$$

# Chapter 3

# Methods

## 3.1 Programming language and packages

The programming language used for this thesis is Python (version 3.6.1). Additionally, the IDEs and packages of Python used are listed in Table 6.1 and Table 6.2 on page 95 in the Appendix.

## 3.2 The data set

The HNC data set consisted of 225 patients, who were identified by a random number between 1 and 230 (patient numbers 20, 45, 70, 72 and 121 were non-existent). For each patient, the data set consisted of four files: two UINT16 files and two CSV-files.

The UINT16 files were of the CT images and photon dose plans. Note that proton dose plans were not included in the data set, and thus were not examined or evaluated.

One of the CSV-files contained location coordinates of OARs, PTVs, ITVs and CTVs drawn in by radiologists. The second CSV-file contained information of the dimensions, resolution and maximum intensity values of the CT images and dose plans. The resolution of all images in the data set was 2 $mm^3$ per voxel, but the dimensions and maximum intensity values differed between each patient.

Before using the data set for feature extraction and analysis, the data set had to be imported correctly and examined for missing data and outliers.

### 3.2.1 Preprocessing the structure data

The structures, meaning the OARs and target volumes, were drawn in by various radiologists. Therefore, labelling was not consistent for the same type of structure. Consequently, a common name for the same type of structure was needed for comparing structures and patients. Also, not all patients had delineated data for all structure types. This is important to know when doing analyses later on.

#### OARs

Three OARs were commonly contoured: the brain stem / medulla, parotid glands, and the submandibular glands. By identifying certain words in the labels given by the radiologists, the label was swapped with the common name chosen for the OAR. For example, if a radiologist's label contained the word 'med' in the string, it implies that the structure identified is the medulla. Hence, the structure would be given the common name for brain stem / medulla.

Additionally, the right and left located salivary glands were distinguished. As left and right were labelled in ten different ways by radiologists, all of these variations needed to be considered. The common name given to right and left were dex and sin, because dex and sin are abbreviations for right and left in Latin.

In rare cases, the word used to identify an OAR was found in more than one label of a delineated volume. To ensure no information was lost, the union of these were given the common structure name. Also, the Python script that fetches the structures data from the CSV-files, was written to manage other specific deviating cases regarding two radiologists and one patient.

Furthermore, if an OAR was not one of the three mentioned OARs, but was described by the radiologist as a gland, this OAR was simply categorised as such. Unidentified glands were contoured for patient numbers 8, 71 and 179. As the identification of these glands was not possible, the data of these were imported, but not used further on in the process.

Patient number 9 was the only patient who lacked data of OARs.

### Target volumes

Although all patients had the tumour drawn in by radiologists as either PTV, ITV or CTV, not all patients had all three. Most of the patients had a combination, where 189 patients had PTV labels, 218 had ITVs, and 22 had CTVs.

In addition to the different labels given to the target volumes by the radiologists, the method of contouring the target volumes also varied. At first glance there appeared to be several target volumes of each type for some patients. For example, a radiologist might have delineated 14 ITVs. However, most of these overlapped, as illustrated in scenario 2 and 3 in Figure 3.1. The reason as to why there are several overlapping volumes delineated is that some radiologists have noted down the necessary dose for each target volume, whereas others have given reasons as to why a certain portion of the volume is an irradiation target. Therefore, it was decided that if two of the same type of target volume overlapped, this was to be considered as one. Nevertheless, some target volumes did not overlap, as illustrated in Figure 3.1 in scenario 1, and here the target volumes were labelled as distinct cases.

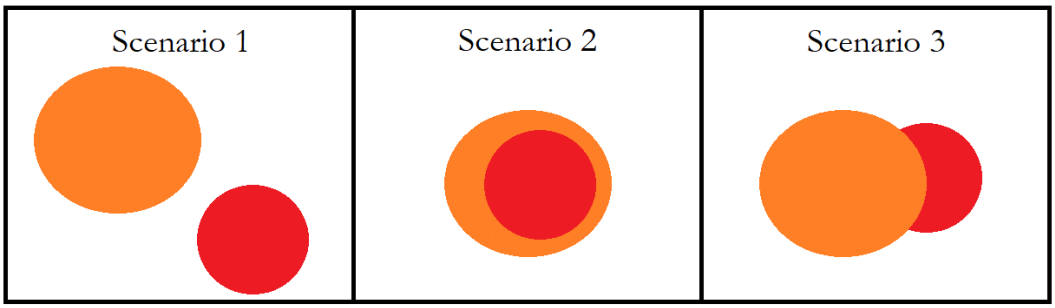



Figure 3.1: Two-dimensional illustration of different target volume contouring. Both the orange and red objects refers to two separate target volumes contoured by a radiologist. In scenario 1 (*left*), the two contours are completely separate target volumes. In scenario 2 (*middle*), and 3 (*right*), there is full or partial overlap between the volumes.

### 3.2.2 Setting up correct image format

The structures were given in IDL coordinates, hence these needed to be transformed into Python coordinates. Interactive Data Language (IDL) is a programming language which structures its coordinate system differently than Python does. In IDL, each voxel has a number referring to a coordinate in the *xyz*-space. Therefore, a hashing function was needed to convert the voxel numbers into Python coordinates. This approach was based on the fact that IDL and Python have different indexing styles, which are illustrated in Figure 3.2.

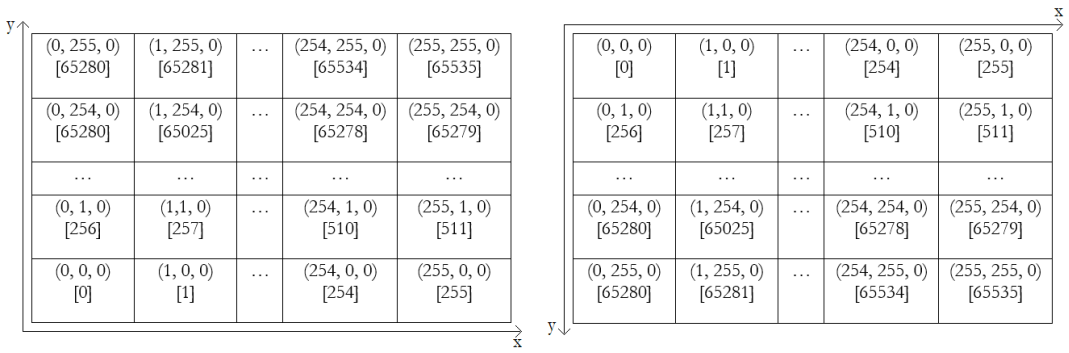



Figure 3.2: Matrix with dimensions $256 \times 256 \times nz$, where $nz$ is the slice number. When converting from IDL (*left*) to Python format (*right*), the voxel numbers need to be converted to the *xyz*- coordinate system used in Python. Coordinates are listed in parentheses and coordinate number in brackets.

When conversion of the arrays was achieved, the structures data was rotated until the CT images, dose plans and structures were aligned with one another in imaging plots. The dose plan, CT image and structures data for patient number 14 is illustrated in Figures 3.3 to 3.9. All images were saved as numpy arrays for easy access later on.


Figure 3.3: Illustration of a CT image for a patient with HNC (patient number 14, axial view: slice 75, sagittal view: slice 125, WL = 100, WW = 70).

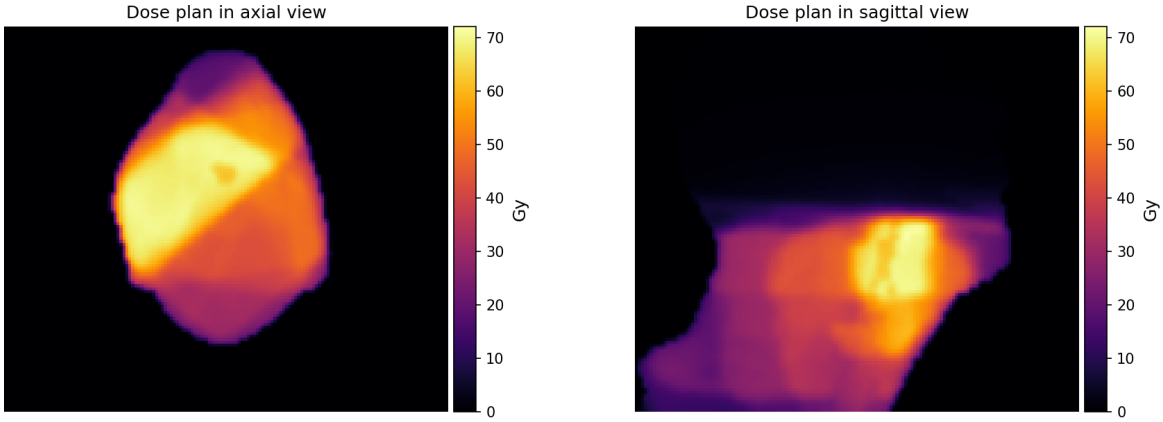



Figure 3.4: Illustration of a dose plan for a patient with HNC (patient number 14, axial view: slice 75, sagittal view: slice 125). The colour bar gives the dose in Gy.

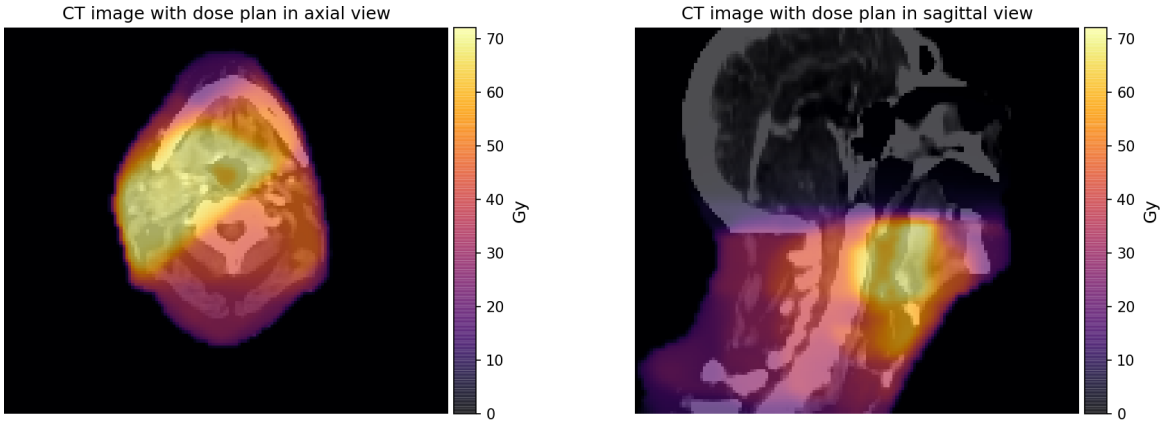



Figure 3.5: Dose plan superimposed on a CT image for a patient with HNC (patient number 14, axial view: slice 75, sagittal view: slice 125, WL = 100, WW = 70). The colour bar gives the dose in Gy.

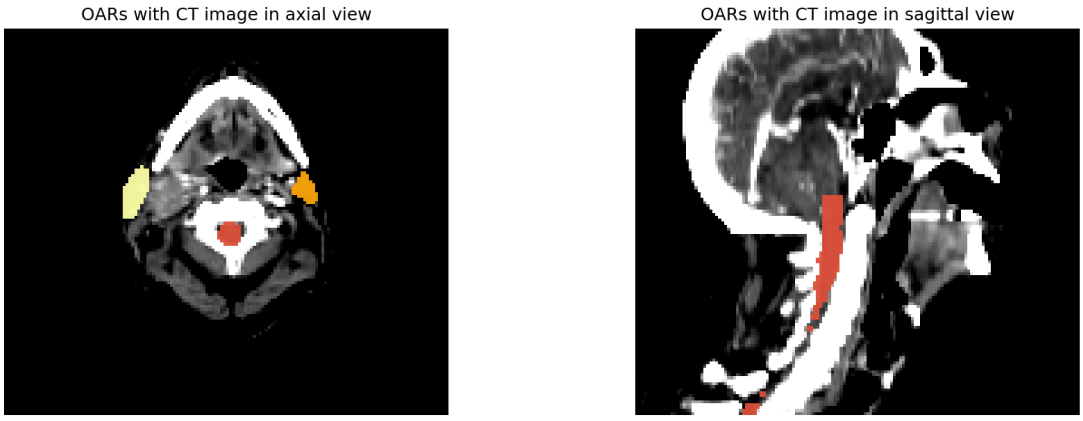



Figure 3.6: OARs superimposed on a CT image for a patient with HNC (patient number 14, axial view: slice 75, sagittal view: slice 125, WL = 100, WW = 70). In the axial view, the parotid glands (yellow and orange) and the brain stem (red) are visible. In the sagittal view, only the brain stem is visible.

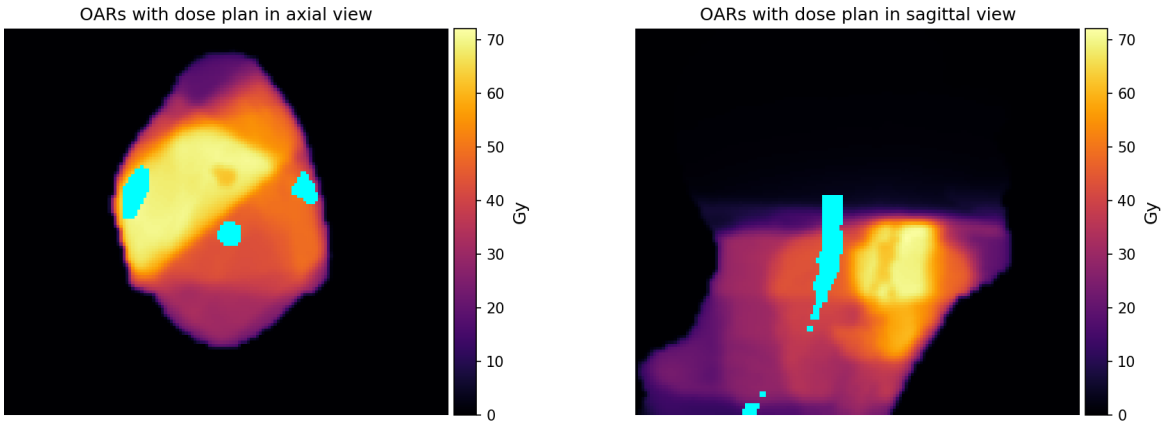



Figure 3.7: The brain stem and parotid glands highlighted in turquoise on the dose plan for a patient with HNC (patient number 14, axial view: slice 75, sagittal view: slice 125). The colour bar gives the dose in Gy.

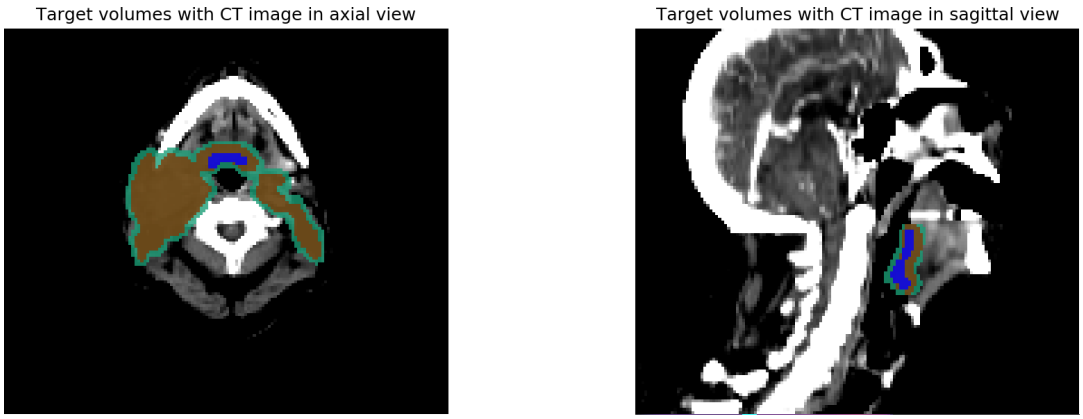



Figure 3.8: Target volumes superimposed on a CT image for a patient with HNC (patient number 14, axial: slice 75, sagittal: slice 125, WL = 100, WW = 70). PTV in green, ITV in brown and CTV in blue.

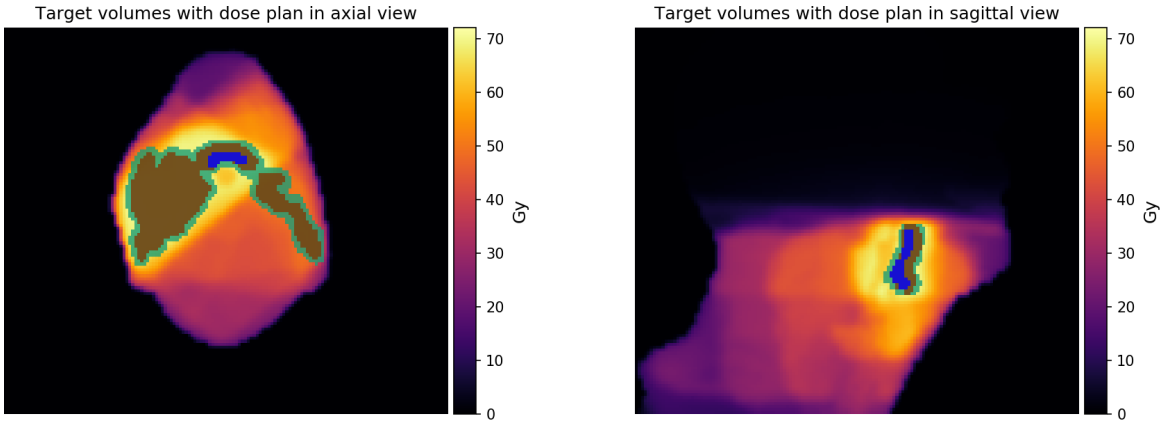



Figure 3.9: Target volumes superimposed on a dose plan for a patient with HNC (patient number 14, axial: slice 75, sagittal: slice 125). PTV in green, ITV in brown and CTV in blue. The colour bar gives the dose in Gy.

## 3.3 Feature extraction

Features extracted for all patients using the CT images, dose plans, OARs and target volumes were:

- Size of PTVs and OARs.
- Mean and median dose for PTVs and OARs.
- Distance between PTVs and OARs using their centre of masses.
- Maximum and minimum distance between PTVs and OARs.
- Weighted distance using CT values (see Section 3.3.1).

The only features extracted that are not explained in Section 2.6 on page 25 or Section 3.3.1 are the mean and median dose of PTVs and OARs. These were extracted by knowing the coordinates of a structure, and then calculating the mean or median value of the dose for the these coordinates.

Additionally, for the OARs that had a minimum distance of zero to a PTV, the overlap volume and Dice were calculated. All features determined were set into dictionaries and saved as pickle-files.

See Appendix for *create_features.ipynb*, which is the Python script for extracting these features.

### 3.3.1 Weighted distance using CT values

As aforementioned in Section 2.4 on page 9, the attenuation coefficient of bones are higher than for soft tissue, and therefore absorb more of the radiation energy [36]. Thus, the type of tissue the x-ray transmits through may affect the dose deposited to the structure.

To take advantage of this information, a weighted distance was created. The purpose of the weighted distance was to represent both the distance and type of tissue existing between a PTV and OAR. As aforementioned, the CT image voxel values can be used to describe the type of tissues at different volumes. Thus, the weighted distance is defined as $sum(CTvalues) * distance$, where $distance$ is the centre distance between the OAR and PTV. $CTvalues$ are voxel values of the CT image for a certain number of voxels that exist on the distance line. The number of CT voxels correspond to the voxel distance between the structures.

## 3.4 Data excluded in analyses

Due to the fact that a PTV describes the irradiated volume more accurately than the other target volumes, the ITVs and CTVs were disregarded for further work in this thesis. This decision was based on the advice from the radiologists and medical physicists at Oslo University Hospital and University of Oslo [14][31].

Further, some patients had more than one PTV (58 patients). For comparison purposes, these patients were excluded in an EDA if the analysis involved the distance features. Thus, the same patients were also excluded from the machine learning approach.

In addition to the criterion of one PTV, a patient needed to have one of the five identified OARs (medulla, parotid dex, parotid sin, submandiblar dex or submandibular sin) when analysing the features of OARs.

## 3.5 Exploratory data analysis approach

The purpose of the EDA was to analyse the correlations between the created features. An important hypothesis was that the dose given to an OAR was correlated with its location in the body and its distance from the PTV. Thus, a correlation matrix for the OARs and their associated features was created. The features used for this correlation matrix was mean dose, median dose, size of OARs, centre distance, weighted distance, maximum distance and minimum distance. In addition, a pairwise plot using the centre distance, weighted distance, mean dose and median dose was created.

All patients with delineated PTVs and OARs (except one patient) had at least one OAR that had a minimum distance of zero. This implies that many OARs overlapped with the PTVs. As a result, it might be difficult to find a correlation using solely minimum distance and dose received. Therefore, a correlation matrix and a pairwise plot between the number of overlapping voxels and the dose given to an OAR were constructed. To further investigate if there was a monotonic relationship, the Spearman correlation was calculated for the same patients.

Results during the EDA illustrated different correlation trends between dose received and overlap volume and Dice for the parotid and submandibular glands. Therefore, these categories were separated into individual correlation matrices. Accordingly, a correlation matrix for the salivary glands glands that overlap with a PTV was created for the features mean dose, median dose, overlap volume and overlap Dice. An equivalent correlation matrix was constructed for the parotid glands. The Spearman coefficient between mean dose and overlap volume for each salivary

gland type was also calculated.

Professor Eirik Malinen informed that the mean dose to a PTV should be 68 Gy, and could be a little less for the PTV borders (60 Gy) [31]. Also, there was a common agreement that the radiation dose to a medulla should be low [14], and Professor Malinen advised that the maximum mean dose should be 48 Gy. To investigate these statements and the dose distributions of the other OARs, box plots and histograms of the mean doses were constructed. For both dose distribution plots, all PTVs, brain stems, parotid glands and submandibular glands of the data set were used.

Professor Malinen also informed that if one salivary gland of a pair was planned to receive a large radiation dose, then the other salivary gland would be spared [31]. To confirm this, histograms illustrating the absolute mean difference in dosage between salivary gland pairs were generated. When constructing these histograms, all patients who had delineated both the left and right-sided parotid or submandibular glands were used.

## 3.6 Machine learning approach

The aim of the machine learning approach was to assess whether an algorithm could predict the mean dose of an OAR based on the features extracted. Both options of classification and regression were tested. Additionally, a Principal Component Analysis (PCA) was performed to compare the classification results with the features and instances of OARs.

The patients with only one distinct PTV were used as data for the machine learning approach. The other instances of PTVs and unidentified glands were not included either.

The features set up for machine learning was minimum distance, maximum distance, centre distance, weighted distance, size of OARs, and the type of OAR the instance was identified as. The options of type of OAR were brain stem / medulla, parotid dex, parotid sin, submandibular dex or submandibular sin. Moreover, the feature describing type of OAR, which was categorical data, was transformed into dummies using the pandas `get_dummies` function.

A parameter was added to the function which fetched the data. If the parameter `merge_glands` was set to `True`, the salivary gland pair names were merged. Thus, the left and right submandibular and parotid glands would not be differentiated, but simply defined as a submandibular or parotid gland. The set-up of the feature matrix when `merge_glands = False` and `merge_glands = True` is illustrated in Figure 3.10 and Figure 3.11 on page 46. Note that when further referring to the feature combination 'all features', the features used are minimum distance, max-

imum distance, centre distance, weighted distance, size, and the type of OAR using `merge_glands = False`.

Additionally, all train, test and validation splits were split using 30 % of the data as test set, and 30 % of the training set for validation. The splits were partitioned by this ratio, because 70:30 is a common split ratio [45], and there was no time to explore for optimal splits. Also, all the partitions were stratified when executing classification to ensure the same distribution of classes in the splits.

The chosen metrics for classification and regression were the *F1* micro-average score and the $R^2$ score. The micro-average was chosen, because each class label would then be weighted equally. For regression, both *RMSE* and $R^2$ are good candidates to use as metrics [69]. However, the $R^2$ was chosen due to the advantage of its higher interpretability.

Note that a random state is set for all `train_test_splits` and algorithms. The random state number is the seed of the pseudo random number generator used when data is shuffled in the function [46]. A fixed random state ensures that the output of a function is the same every time the algorithm is executed.

| | Minimum distance | Maximum distance | Centre distance | Weighted distance | Size | Type of OAR_hjst_medulla_spin | Type of OAR_parotid dex | Type of OAR_parotid sin | Type of OAR_submand dex | Type of OAR_submand sin |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 79 | 24 | 485886 | 4084 | 1 | 0 | 0 | 0 | 0 |
| **1** | 11 | 98 | 50 | 2615576 | 4563 | 0 | 1 | 0 | 0 | 0 |
| **2** | 0 | 77 | 26 | 532639 | 4973 | 0 | 0 | 1 | 0 | 0 |
| **3** | 9 | 67 | 34 | 836041 | 1067 | 0 | 0 | 0 | 1 | 0 |
| **4** | 0 | 51 | 14 | 182913 | 1332 | 0 | 0 | 0 | 0 | 1 |

Figure 3.10: Five first instances of the feature matrix when `merge_glands = False`.
Note that the left (sin) and right (dex) parotid and submandibular salivary glands are specified.

| | Minimum distance | Maximum distance | Centre distance | Weighted distance | Size | Type of OAR_hjst_medulla_spin | Type of OAR_parotid | Type of OAR_submand |
|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 79 | 24 | 485886 | 4084 | 1 | 0 | 0 |
| **1** | 11 | 98 | 50 | 2615576 | 4563 | 0 | 1 | 0 |
| **2** | 0 | 77 | 26 | 532639 | 4973 | 0 | 1 | 0 |
| **3** | 9 | 67 | 34 | 836041 | 1067 | 0 | 0 | 1 |
| **4** | 0 | 51 | 14 | 182913 | 1332 | 0 | 0 | 1 |

Figure 3.11: Five first instance of the feature matrix when `merge_glands = True`.

### 3.6.1 Classification

#### Class definitions

The mean dose for the OARs are continuous variables. Therefore, to implement a classification algorithm, the mean dose values had to be categorised. The number of categories set up should reflect the varied dose distribution for all structures. Thus, six categories were defined. These were simply named by a number between 1 to 6. Further, 60 Gy was the lowest dose to a PTV (ergo, qualifies as a high dose), so $>60$ Gy was set as the largest dose category. Table 3.1 explains how the continuous mean dose values were categorised. Hence, the mean dose categories became the target labels for classification.

Table 3.1: Categories of the mean dose values.

| Category number | Dose interval |
|---|---|
| 1 | $\leq 12$ Gy |
| 2 | $< 12$ and $\leq 24$ |
| 3 | $< 24$ and $\leq 36$ |
| 4 | $< 36$ and $\leq 48$ |
| 5 | $< 48$ and $\leq 60$ |
| 6 | $> 60$ |

#### Initial classification and feature selection

After fetching the feature matrix and associated target labels (`merge_glands = False`), the data was split into training and test data. The sets were stratified and test set was 0.3 of the data (random state = 42).

A Random Forest was chosen for an initial prediction trial, because the algorithm is simple and has been proven to work quite well for a large variety of data sets [70]. Therefore, a Random Forest with 12 decision trees (random state = 1) was fitted to the training set, and both training and test sets were predicted with the fitted Random Forest. For illustrating the results, confusion matrices of the train and test sets were created, in addition to calculating each sets' *F1* micro-average score. A histogram of the absolute difference between the true and predicted label for the test set was also constructed.

Next, two Sequential Backward Floating Selection (SBFS) processes were executed on the data set to explore whether feature reduction was possible. The difference between the SBFS processes was that one of the data sets was with (`merge_glands = False`) and the other with (`merge_glands = True`). Also, the number of features requested from the SBFS with `merge_glands = False` was three, and one

feature was requested for the other SBFS. The SBFS processes used the same Random Forest classifier as described, with CV of 5 and estimating the *F1* micro-average score.

### Classifier comparisons

Six different classifiers were evaluated using cross validation with 10 folds. The classifiers tested were Logistic Regression, Decision Trees, KNN, Random Forest, SVM and AdaBoost (all with random state = 1, and the rest of the parameters were the default values). The data was scaled using the `StandardScaler` for Logistic Regression, KNN and SVM. The metric used was *F1* micro-average.

Five combinations of features were tested for each evaluated classifier. The first feature combination used all the features. Additionally, the feature combination with highest *F1* micro-average score of the SBFS with `merge_glands = True` was tested. Further, the same feature combination but without the features regarding OAR type, was attempted (minimum distance, centre distance, weighted distance and size). The reason for removing the OAR dummy features was to check the impact of the OARs on the *F1* micro-average score.

The two feature combinations containing all features except the size feature and all features except centre and weighted distance were also evaluated. The purpose was to evaluate the importance of the removed features on the prediction of mean dose. Centre distance and weighted distance were removed simultaneously because these correlated (see Section 4.1 on page 51).

The feature data sets with different feature combinations were all split into training and test data with test size 0.3 and stratified (random state = 42). Then the training sets were used for estimating each classifier.

### Hyperparameter tuning and evaluation of the highest performing classifier

The training data that contained all features (`merge_glands = False`) was split up to create a new training and a validation set (random state = 34). The validation size was 0.3 of the original training data, and both sets were stratified by the original training data's target values.

Using Logistic Regression (random state = 1, other parameters set to default), the parameter $C$ was tuned using grid search with 10 CVs and the *F1* micro-average score. The values of $C$ tested were 0.0001, 0.001, 0.01, 0.1, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 15.0, 20.0, 100.0, and 1000.0. Standardisation of the validation data was also done prior to the grid search.

Next, the training set was used to fit a Logistic Regression with $C = 5.0$ (random state = 2). The *F1* micro-average score for prediction of both the training and test set were estimated, and associated confusion matrices were constructed. In addition, a histogram of the absolute difference between the true and predicted label for the test set was created.

An identical split of train and validation set, grid search and fitting to a Logistic Regression was done with a data set where the size feature was removed. The only difference in the procedure this time was with the final Logistic Regression. The parameter $C$ was set to 4.0 instead of 5.0 due to different results from the grid search. Thus, the *F1* micro-average score was calculated for the predictions of both the training and test set.

Note: The default method of the sci-kit learn package for converting a Logistic Regression model to a multi-class problem is by OvR. Therefore, OvR was always used for Logistic Regression.

### 3.6.2 PCA

A PCA model was computed for the test set from Section 3.6.1 that contained all features. The PCA had 10 PCs and a 5 fold CV. The results of the PCA were illustrated in a scores plot, loadings plot, correlation loadings plot, and biplot for the two PCs. An explained variance plot was also created.

In addition, a colour coded scores plot was constructed. The colours of the scores were based on the prediction accuracy of the Logistic Regression using all features ($C = 5.0$).

### 3.6.3 Regression

#### Initial regression and feature selection

Firstly, the feature matrix with all features (`merge_glands = False`) and the continuous response variables (mean dose) were fetched and partitioned into training and test data. The sets were stratified and test set was 30 % of the data (random state = 42). Then, the training data was fitted to a Random Forest with 12 decision trees (random state = 1, other parameters set to default). $R^2$ and *RMSE* for predicted mean dose values of the training and test sets were estimated. Additionally, a plot demonstrating the distribution of absolute difference between true and predicted mean dose value of the test set was constructed.

The same Random Forest was used for two SBFS processes to explore options of feature reduction, and to compare with previous SBFS processes. The SBFS processes for classification was done with both `merge_glands = False` and `merge_glands = True` data sets. Therefore, this was automatically done for regression also. The requested number of features for the SBFS with data set `merge_glands = False` was three, and one for the `merge_glands = True`. Moreover, the SBFS processes used 5 fold CV and estimated the score using the average $R^2$ score.

### Regressor comparisons

Regressors were tested using the feature combination which gave the highest average $R^2$ score of the SBFS processes. In addition, the same feature combination minus the OAR dummy features was another feature combination used.

The data set for comparing regressors was with `merge_glands = True`, and the training and test partitioned with a 70:30 split (random state = 42).

The regressors evaluated were Random Forest, Decision Trees, LASSO, SVM, and AdaBoost (all random state = 1, except SVM, and the other parameters set to default). In addition, a data set with quadratic polynomial features fitted to a Random Forest was tested.

### Hyperparameter tuning and evaluation of highest performing regressor

Tuning the regressor with the feature combination that gave the largest $R^2$ score was done next. The validation set was 0.3 of the original training set (random state = 66). Also, quadratic polynomial features were retrieved before the grid search.

The number of decision trees tested for the Random Forest (random state = 10) in the grid search was 10, 20, 40, 70, and 100. Three different approaches to deciding the number of features to consider when evaluating splits were tested. The three approaches were (1) using all features of the data set, (2) the square root of the original number of features, or (3) the logarithm to the base of 2 of the original number of features. Finally, *MSE* and *MAE* were evaluated as performance measurements in the grid search.

A grid search using the validation set and a 10 fold CV with $R^2$ as the metric was implemented. From the grid search, the hyperparameter combination with highest score was used further in training and testing a Random Forest (random state = 2). Training with the train set and predicting the mean dose of both train and test set was done with and without quadratic polynomial features.

# Chapter 4

# Results

## 4.1 Exploratory Data Analysis on the HNC data set

The strongest linear correlations found between the features given in the correlation matrix in Figure 4.1 on the following page are between the mean and median dose, and the distance between centres and the weighted distance. These were expected to correlate, especially the distances, since the distance between centres is used to create the weighted distance feature.

Furthermore, Figure 4.1 implies a moderate to strong inverse linear correlation between the features centre distance, weighted distance and maximum distance, and mean and median dose. As a result, the hypothesis of distance affecting the dose given to the OARs is confirmed. However, to further confirm the linear dependency, a pairwise plot was constructed, shown in Figure 4.2 on page 53.

The comparative scatter plots in Figure 4.2 illustrate a somewhat linear correlation between dose to OAR and distance from PTV, yet the spread of points is too large to see a clear linear correlation. Nonetheless, the pairwise plot confirms a slight linear dependency. Thus, there are probably other factors than solely distance that affect the dose given to an OAR.

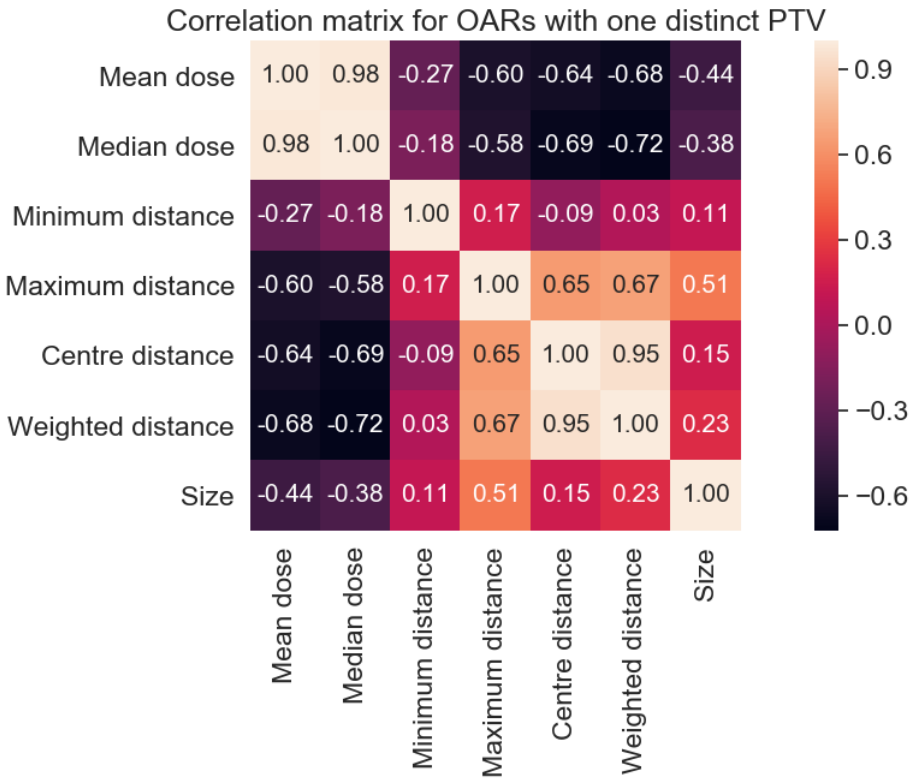



Figure 4.1: Pearson correlation matrix for OARs with patients that had only one PTV (130 patients). The features compared are the sizes of OARs, mean and median dose to the OAR, and the distance between mass centres, weighted distance, and maximum and minimum distance (all between OARs and PTV).

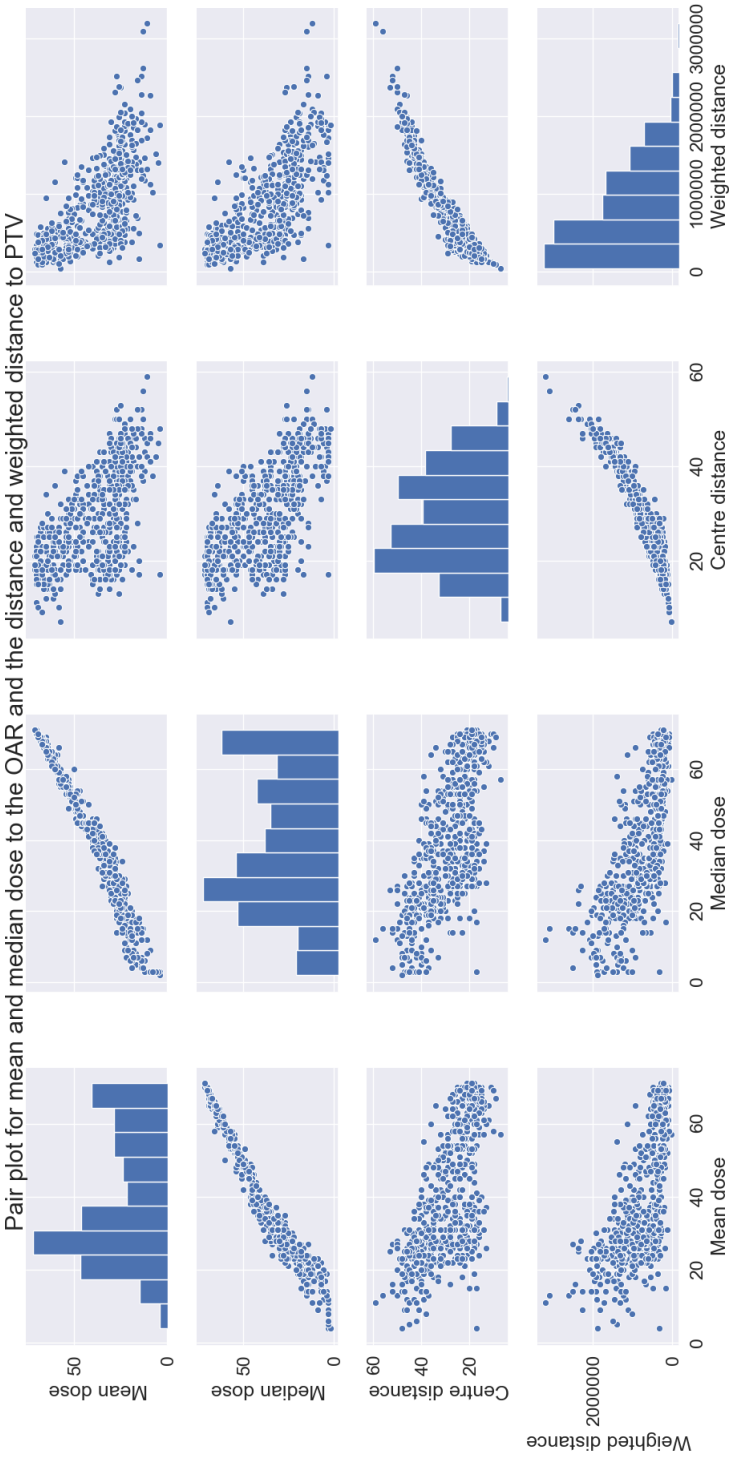



Figure 4.2: Pairwise plots for OARs from patients that had only one PTV. The comparisons were done using the features mean and median dose given to the OAR, and the centre distance and weighted distance between OAR and PTV. The histograms along the diagonal are the distributions of the features for the OARs.

The correlation matrix in Figure 4.3 illustrates a moderate linear correlation between the number of overlapping voxels and Dice, and dose received. A correlation between the overlap with PTV and dose received by the OAR might be observed in the scatter plots of Figure 4.4 on the next page. However, because the data set has a dose limit, the trend observed in Figure 4.4 might not be linear.

The Spearman's correlation was calculated using all patients with delineated PTVs and OARs that overlapped. The result was a $r_s = 0.45$, implying a moderate, monotonic correlation between the number of overlapping voxels and dose received.

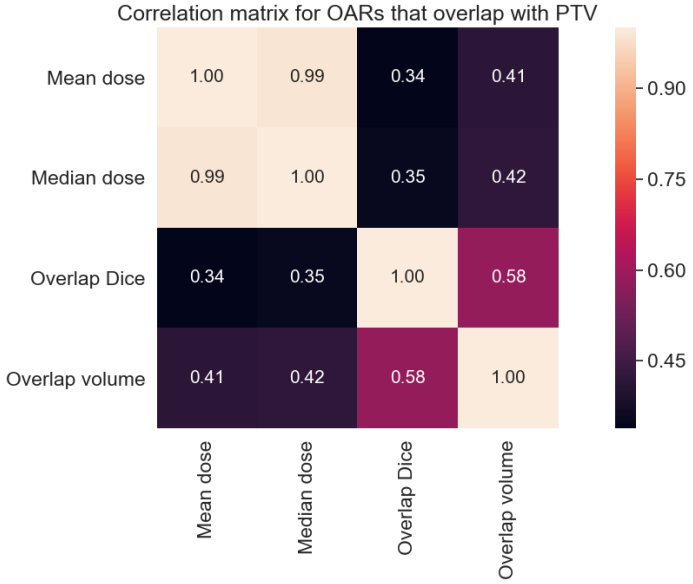



Figure 4.3: Pearson correlation matrix for OARs that overlap with the PTV. The features compared are the mean and median dose received by the OARs, overlap volume and overlap Dice between OAR and PTV.

The correlation matrices of overlap metrics for the submandibular and parotid glands are in Figure 4.5 and Figure 4.6 on page 56, respectively. The linear correlation between the dose received and the overlap volume was larger when the submandibular and parotid glands were examined separately in Figure 4.5 and Figure 4.6 than when all three OARs were used in Figure 4.3. Hence, a moderate linear correlation was found between the dose received and overlap volume for the submandibular gland and parotid glands separately. Moreover, Spearman's coefficient between the mean dose and overlap volume was 0.63 and 0.57 for the submandibular glands and parotid glands, respectively. This implies a somewhat stronger monotonic than linear relationship for the submandibular glands.

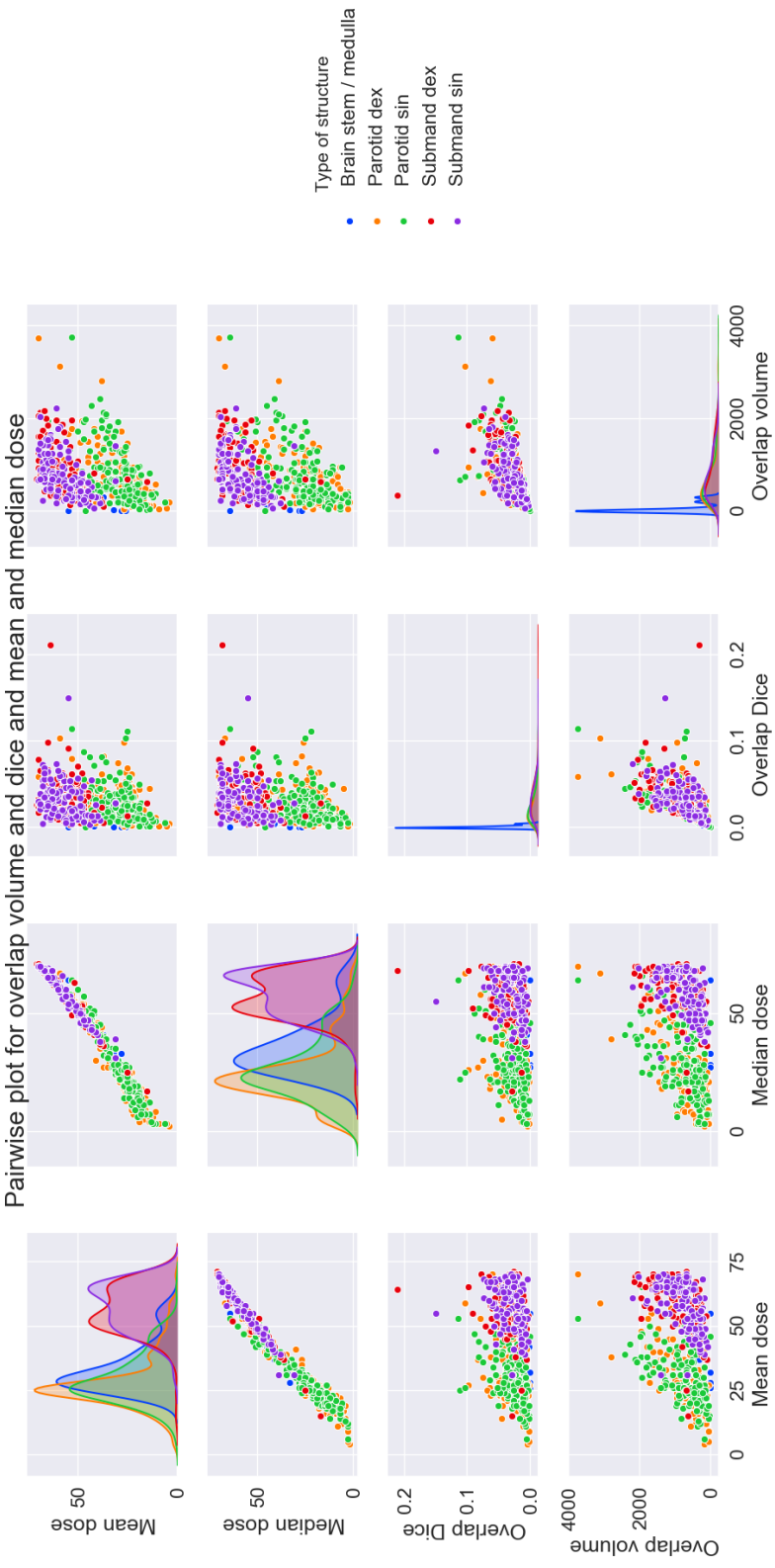



Figure 4.4: Pairwise plots between mean dose, median dose and the number of overlapping voxels between the OARs and PTVs. The type of OAR is highlighted according to the colours in the legend on the right. The plots along the diagonal are averaged histograms of the OARs' dose and overlap distributions.

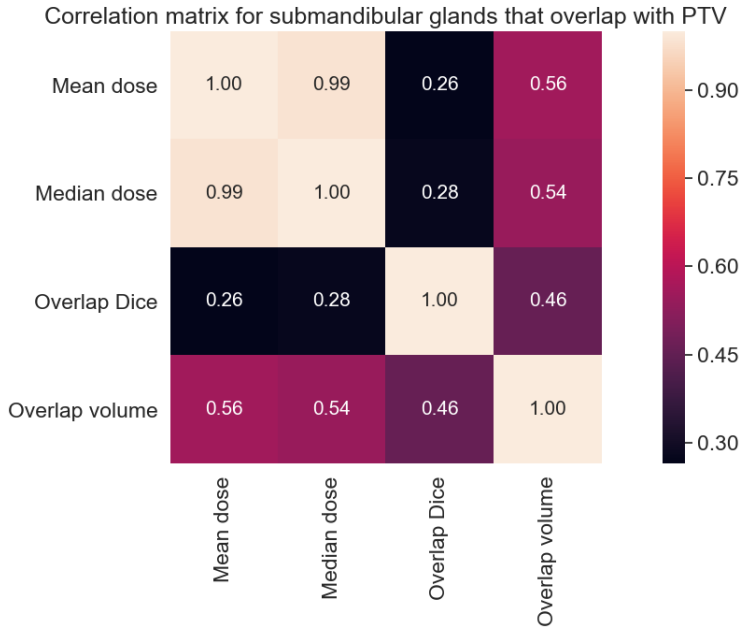



Figure 4.5: Pearson correlation matrix for submandibular glands that overlap with PTV. The features compared are the mean and median dose to the submandibular glands, overlap volume and overlap Dice.

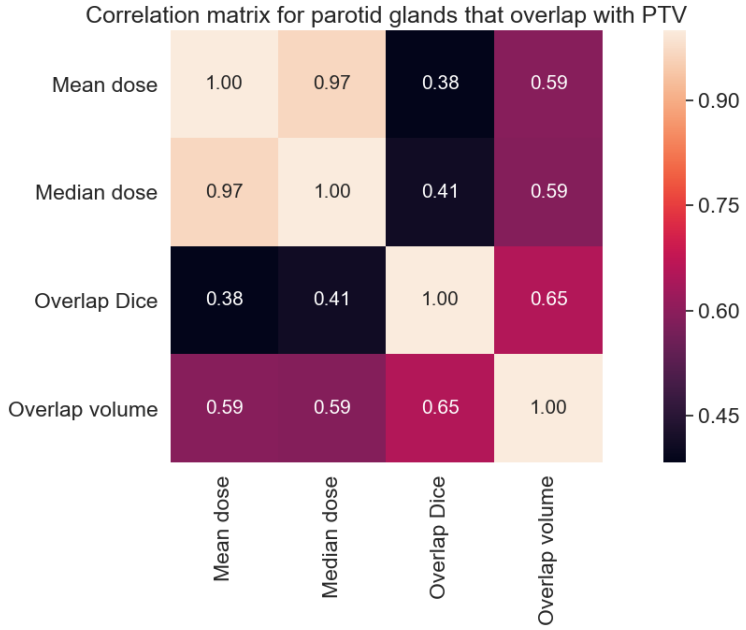



Figure 4.6: Pearson correlation matrix for parotid glands that overlap with PTV. The features compared are the mean and median dose to the parotid glands, overlap volume and overlap Dice.

Dose distributions received by the structures delineated by radiologists were also investigated. Figures on page 58 and 59 are box plots and histograms of the mean dose distributions for the brain stem / medulla, parotid glands, submandibular glands and PTVs. These figures show that the submandibular and parotid gland pairs had similar dose distributions. Further, the mean dose to the medulla was never above 48 Gy, except in one patient. Also, as expected, the mean dose given to the PTVs was usually large. However, some PTVs received a mean dose lower than 60 Gy.

The absolute mean dose difference distributions between gland pairs were plotted. The histograms in Figure 4.9 on page 60 show that most salivary gland pairs differ by less than 10 Gy. Thus, salivary gland pairs often received about the same radiation dose.

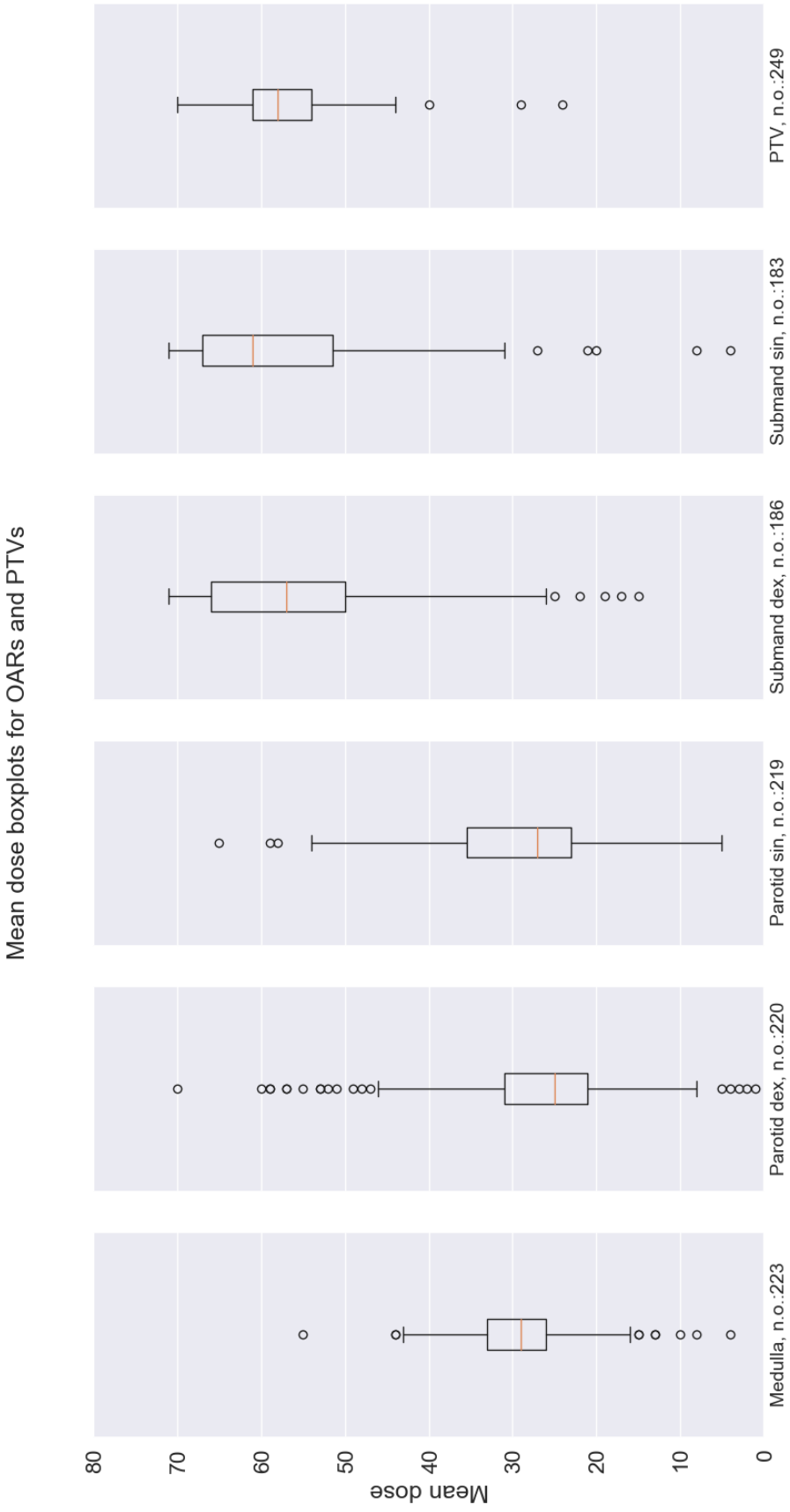



Figure 4.7: Box plots for mean dose given to patients' medulla, parotid glands, submandibular glands, and PTV. Since not all OARs have been delineated in every patient, the number OARs that contributed to each box plot is noted below. All delineated PTVs contributed to the PTV box plot.

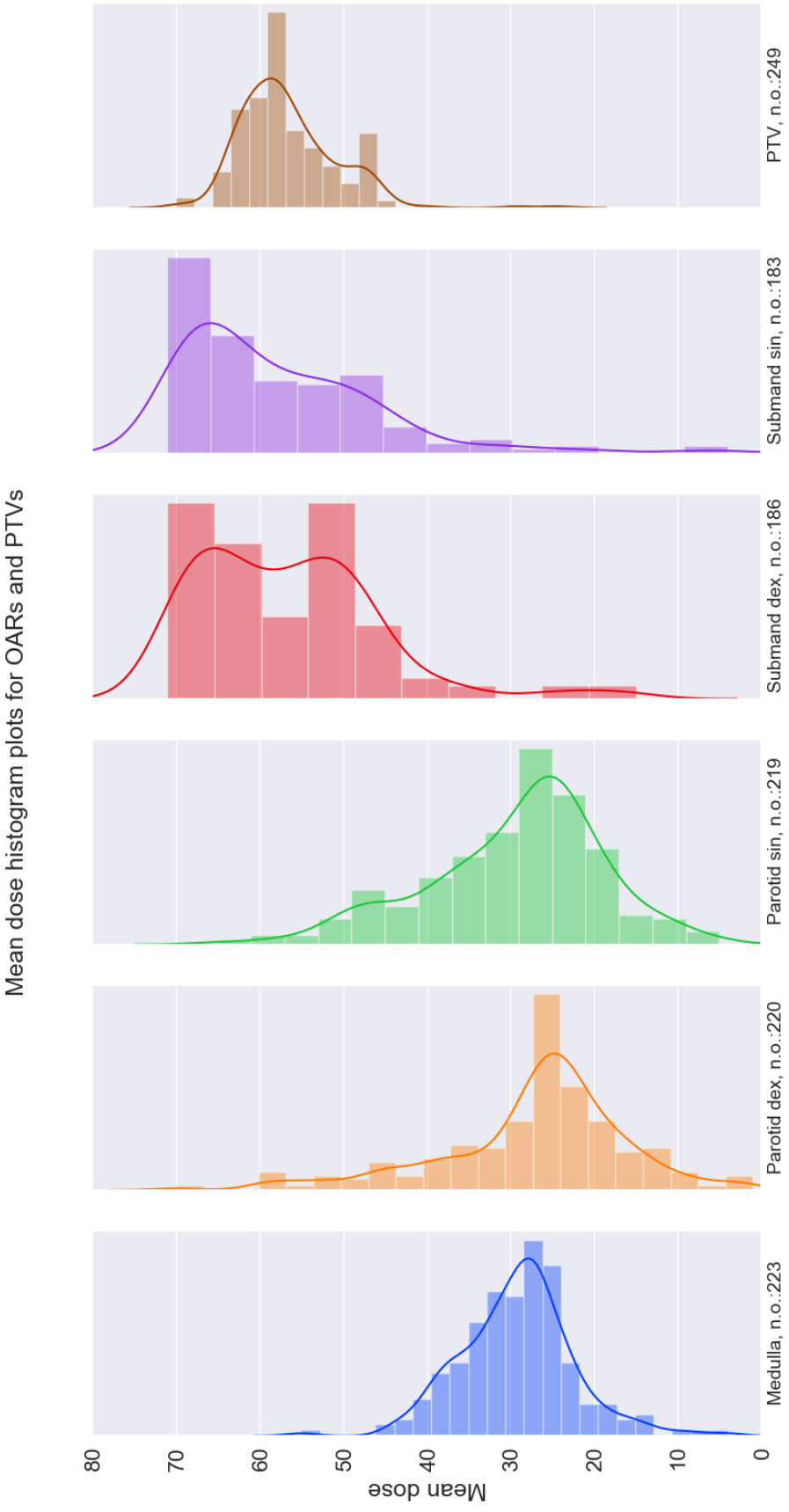



Figure 4.8: Histogram distributions for mean dose given to patients' medulla, parotid glands, submandibular glands, and PTV. Since not all OARs have been delineated in every patient, the number OARs that contributed to each histogram is noted below. All delineated PTVs contributed to the PTV histogram distribution.

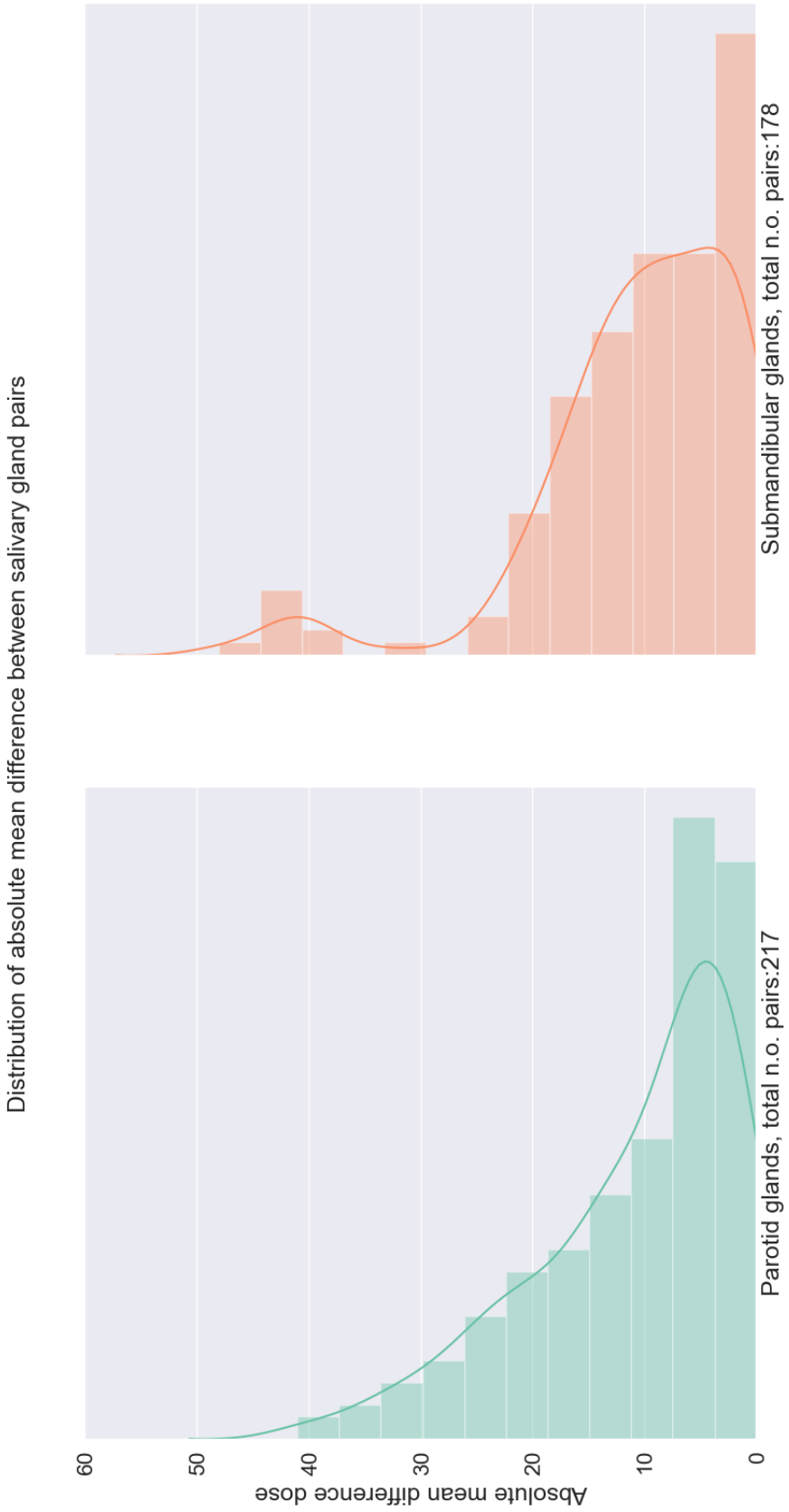



Figure 4.9: Absolute mean difference dose between parotid and submandibular gland pairs.

## 4.2 Machine learning

### 4.2.1 Classification

#### Initial classification

Classification of OARs into dose classes gave *F1* micro-average scores of the train and test sets for Random Forest of 0.988 and 0.511, respectively. Thus, this model suffered from overfitting, because there is a large difference between the train and test score. The confusion matrices of the train and test set in Figure 4.10 and Figure 4.11 on the following page also demonstrates overfitting.

Almost all instances were categorised correctly in the train set, whereas not for the instances of the test set. However, the confusion matrix of the test set shows a trend of correctly categorised instances. Moreover, few of them were predicted incorrectly by more than one difference in category.

Similar findings are reflected in Figure 4.12 on the next page. Here, almost all of the incorrectly labelled instances were placed in the bin which reads '1', meaning the absolute difference between true and predicted label for these instances was 1. Figure 4.12 also demonstrates that about half of the instances were predicted incorrectly.

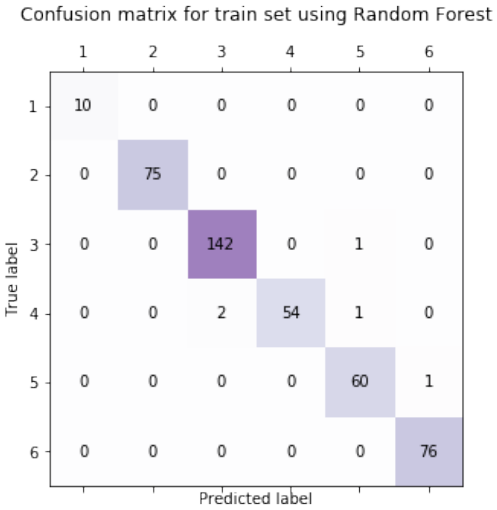



Figure 4.10: Confusion matrix for the train set when using a Random Forest with 12 decision trees.

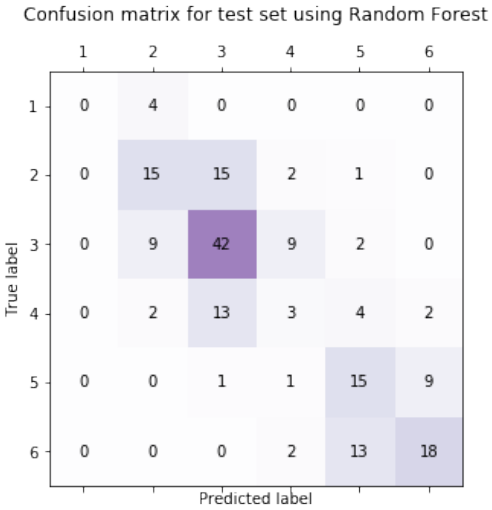



Figure 4.11: Confusion matrix for the test set when using a Random Forest with 12 decision trees.

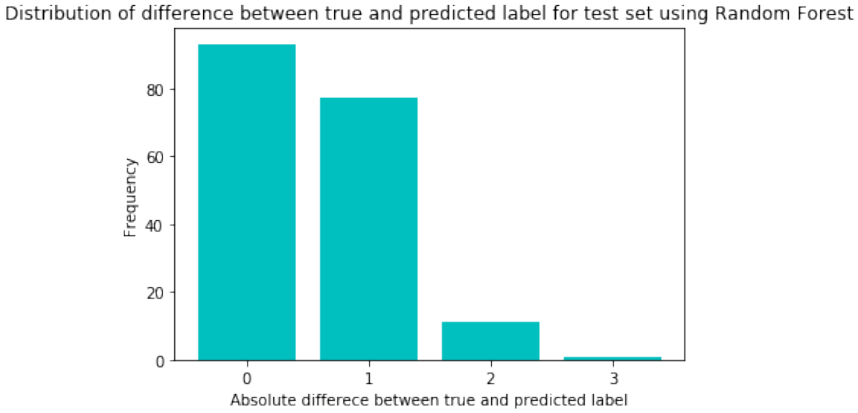



Figure 4.12: Histogram for the distribution of absolute the difference between the true and predicted label for the test set using Random Forest with 12 decision trees.

### Feature selection using Sequential Backward Floating Selection

The result of the Sequential Backward Floating Selection (SBFS) process when `merge_glands = False` is illustrated in Figure 4.13 on the following page. The final combination of three features from the SBFS was the minimum distance, maximum distance and centre distance.

The highest *F1* micro-average score obtained when `merge_glands = False` included the features describing the left salivary glands; parotid sin and submandibular sin. This might suggest that the type of salivary gland affects the score. However, whether a salivary gland is on the left or right side should not be a factor to the score. Hence, another SBFS was done with `merge_glands = True` to prevent the algorithm having to choose between left or right salivary gland. The result of the second SBFS is in Figure 4.14 on page 65.

According to the second SBFS shown in Figure 4.14, the dummy feature of the submandibular glands was the feature which alone could give the highest *F1* micro-average score. In Figure 4.13, however, the left and right submandibular gland features were not prioritised as highly. Therefore, this is an indication that merging the left and right salivary gland labels had an impact on the score.

In both SBFS processes, the size feature, which described the size of the OARs, was removed early and did not make a reappearance. Thus, the SBFS indicates that size is not essential for prediction. On the other hand, the removal of either parotid or parotid dex feature (depending on which SBFS) decrease the model's score. Therefore, the SBFS processes suggest that the dummy feature of parotid glands contained relevant information for the mean dose prediction.

It is difficult to map how the weighted distance and medulla impact the prediction accuracy. Nevertheless, it is plausible that both contribute to some information, because the score decreased when these features were removed.

Regardless, both SBFS processes showed a minimal change in performance for all feature combinations shown in Figure 4.13 and Figure 4.14. Thus, whichever feature combination chosen for an optimal model later might not have a large influence on the score.

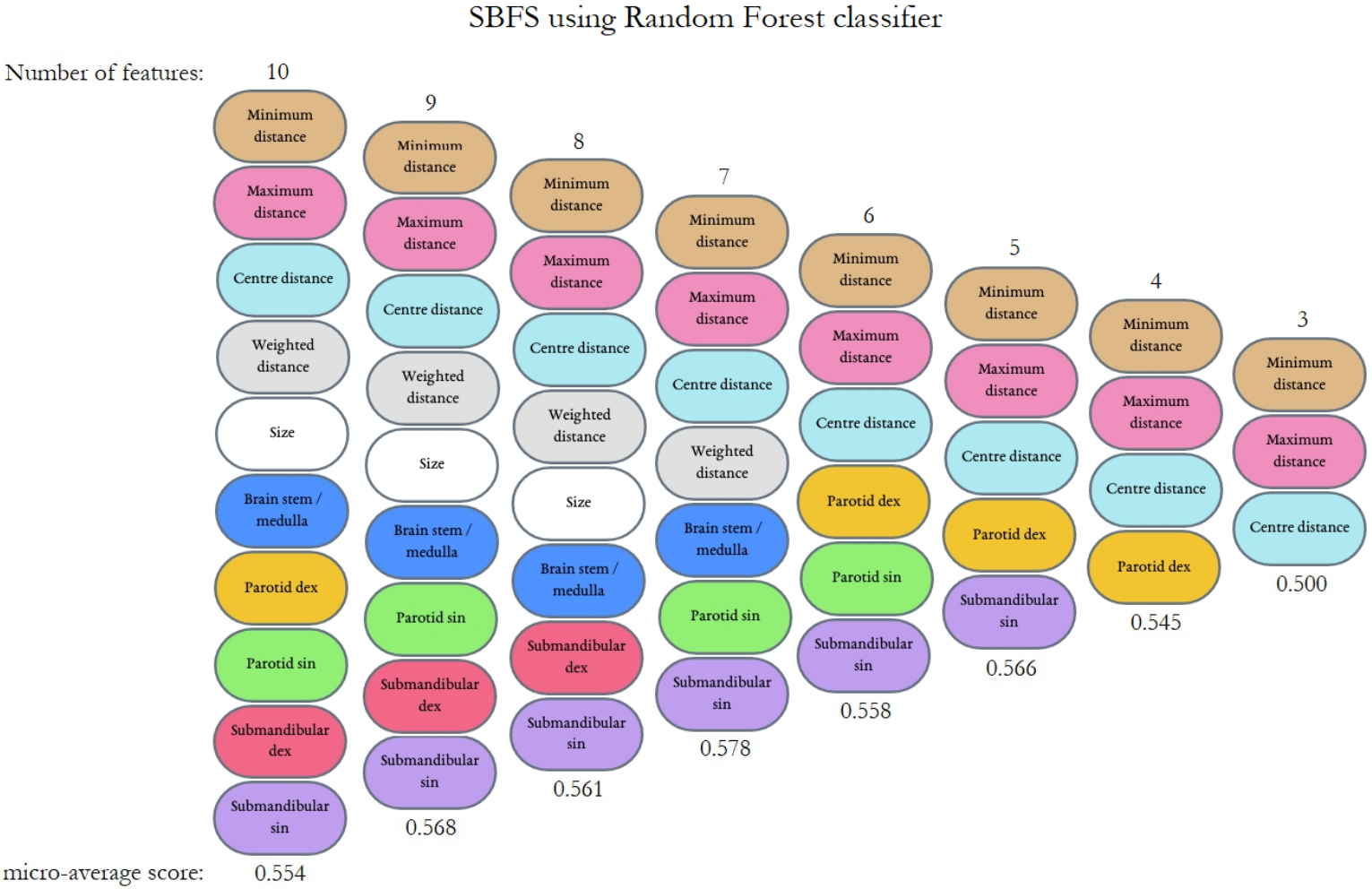



Figure 4.13: The SBFS result for classification when `merge_glands = False`. The far-left column represents the first iteration where the SBFS used all available features. The far-right column illustrates the three features the SBFS decided to keep. The columns from left to right represents a feature being removed for each iteration step of the SBFS. Additionally, the features are depicted with a specific colour. The number of features is written above each iteration column, and the *F1* micro-average score is noted below.

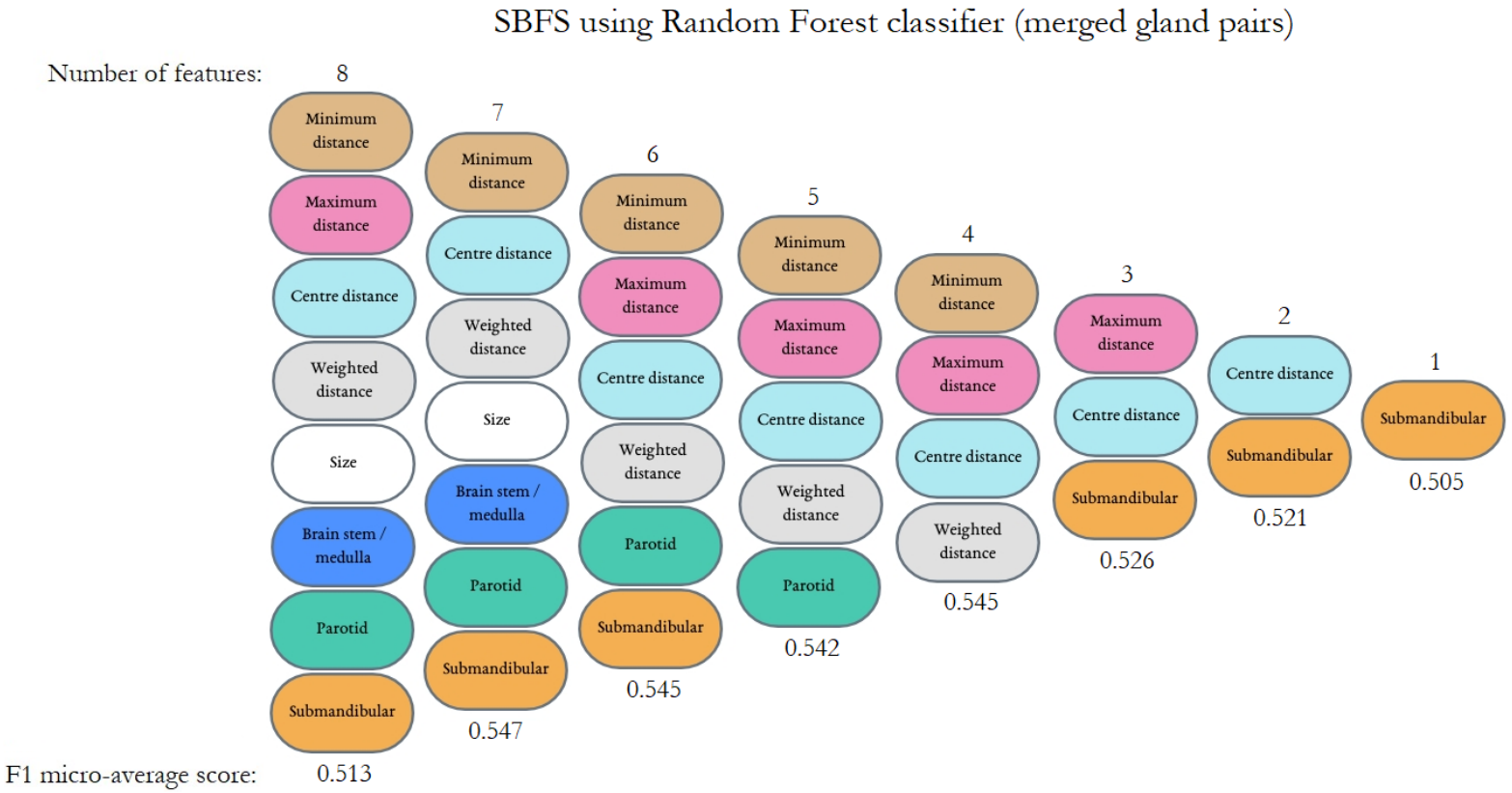



Figure 4.14: The SBFS result for classification when `merge_glands = True`. The far-left column represents the first iteration where the SBFS used all available features. The far-right column illustrates the one feature the SBFS decided to keep. The columns from left to right represents a feature being removed for each iteration step of the SBFS. Additionally, the features are depicted with a specific colour. The number of features are written above each iteration column, and the *F1* micro-average score is noted below.

### Classifier comparisons

Table 4.1 on the facing page is an overview of the algorithms' *F1* micro-average score using different feature combinations.

When the features centre distance and weighted distance were removed, all classifiers achieved a lower *F1* micro-average score. Thus, these two features combined contain relevant information to the prediction of mean dose to and OAR.

The highest scoring feature combination of the SBFS when `merge_glands = True` was with the features minimum distance, centre distance, weighted distance, size of OARs, brain stem / medulla, parotid and submandibular. Additionally to this feature combination, only using the features minimum distance, centre distance, weighted distance and size was also tested on different classifiers. However, there was no agreement between the classifiers of which of these two feature combinations resulted in a higher model performance. Therefore, the results in Table 4.1 gave no indication as to what impact the dummy features had on the prediction score.

From Table 4.1, the Logistic Regression algorithm had the overall highest score, with 0.59 as the greatest. Thus, Logistic Regression was chosen as the algorithm to tune. Further, the score of Logistic Regression was 0.59 for the feature combinations 'All features' and 'All features except Size'. Also, the standard deviation difference was only 0.001. Thus, both feature combinations were used for tweaking a Logistic Regression.

Table 4.1: Result of cross validating different classifier algorithms using the *F1* micro-average score (+/- standard deviation of the CV scores). Featuere combinations used are list in the far-left column, and the names of the algorithms used are in the top row. The only instance containing merged pair glands is the feature combination of Minimum distance, Centre distance, Weighted distance, Size, Brain stem / medulla, Parotid, Submandibular.

| | **Logistic Regression** | **Decision Trees** | **KNN** | **Random Forest** | **SVC** | **AdaBoost** |
|---|---|---|---|---|---|---|
| All features | 0.59 (+/- 0.05) | 0.48 (+/- 0.063) | 0.54 (+/- 0.066) | 0.57 (+/- 0.055) | 0.54 (+/- 0.042) | 0.48 (+/- 0.04) |
| Minimum distance, Centre distance, Weighted distance, Size, Brain stem / medulla, Parotid, Submandibular | 0.57 (+/- 0.048) | 0.48 (+/- 0.083) | 0.5 (+/- 0.057) | 0.49 (+/- 0.087) | 0.54 (+/- 0.046) | 0.49 (+/- 0.052) |
| Minimum distance, Centre distance, Weighted distance, Size | 0.54 (+/- 0.065) | 0.47 (+/- 0.1) | 0.5 (+/- 0.07) | 0.51 (+/- 0.082) | 0.55 (+/- 0.055) | 0.48 (+/- 0.036) |
| All features except Size | 0.59 (+/- 0.051) | 0.5 (+/- 0.049) | 0.55 (+/- 0.075) | 0.53 (+/- 0.05) | 0.54 (+/- 0.052) | 0.45 (+/- 0.044) |
| All features except Weighted distance and Centre distance | 0.51 (+/- 0.025) | 0.39 (+/- 0.069) | 0.46 (+/- 0.079) | 0.43 (+/- 0.068) | 0.5 (+/- 0.022) | 0.35 (+/- 0.071) |

**Hyperparameter tuning and evaluation of the highest performing classifier**

The grid search using all features led to the optimal value of $C = 5.0$. Thus, by using Logistic ($C = 5.0$) with all features (`merge_glands = False`), the *F1* micro-average score for the train and test sets were 0.61 and 0.58, respectively. The confusion matrices of the train and test sets are in Figure 4.15 and Figure 4.16 on the next page. Additionally, the distribution of difference between the true and predicted label categories of the test set is plotted in Figure 4.17.

Comparing the confusion matrices provided by the Logistic Regression, one can see that there was less overfitting than for the previously fitted Random Forest (Figure 4.10 and Figure 4.11 on page 62). Yet, by simply comparing the two confusion matrices for the test set (Figure 4.16 and Figure 4.11), it is difficult to observe a significant improvement. With such a small test set, individual instances have a large impact on the score. Therefore, the distribution plots in Figure 4.12 and Figure 4.17 also appear quite similar.

Finally, a Logistic Regression for the data containing all features except size was tuned. The scores of the train and test sets of the Logistic Regression ($C = 4.0$) were 0.62 and 0.58, respectively. Thereby, excluding the size does not improve the test score, and excluding the feature might reduce overfitting.

Both Logistic Regression models with or without the size feature suffered from underfitting. Therefore, the classification models failed to achieve a *F1* micro-average score from both the training and test sets that reflect a high prediction score.

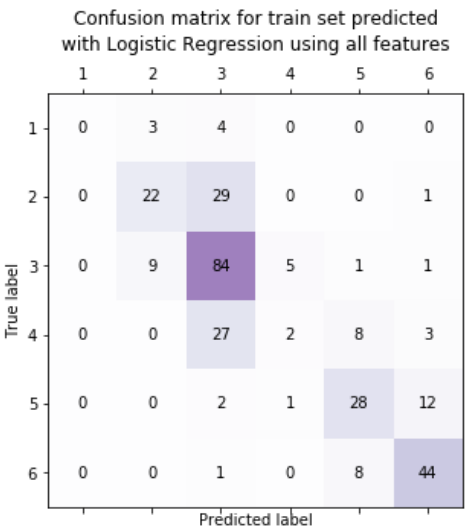



Figure 4.15: Confusion matrix for the train set when using a Logistic Regression ($C = 5.0$).

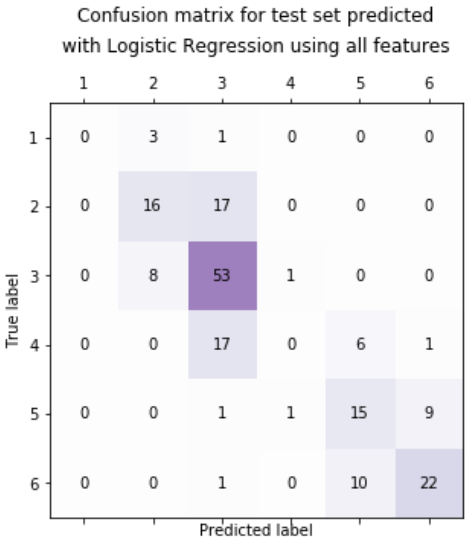



Figure 4.16: Confusion matrix for the test set when using a Logistic Regression ($C = 5.0$).

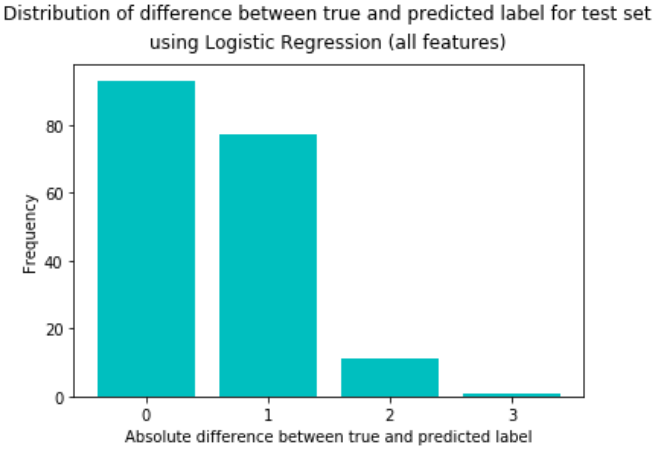



Figure 4.17: Histogram for the distribution of absolute difference between true and predicted label for the test set using Logistic Regression ($C = 5.0$).

### 4.2.2 PCA

Figure 4.18 on page 72 contains the scores plot, loadings plot, biplot and correlation loadings plot of the test set used for classification. The biplot illustrates that the three clusters of scores each correlate with a distinct OAR type. Therefore, PC1 and PC2 are able to distinguish between the different types of OAR. Although, from the correlation loadings plot, the medulla is explained almost 100 % by PC2, whereas the salivary glands are explained less than 50 % by the first two PCs.

Also expressed by the correlation loadings plot, the centre distance, weighted distance and maximum distance are well represented by PC1. Additionally, centre and weighted distance are near each other in the loadings plot, and therefore highly correlated, which was asserted with the EDA results using Figure 4.1 on page 52. In addition, PC2 seems to explain the variance of both minimum distance and medulla, which correlate the most with each other. The loadings plot also demonstrate a high correlation between the left and right-sided parotid salivary glands, as expected.

The score clusters also positively or negatively correlates with the spatial features according to the positioning in the biplot. The top score cluster is positioned close to the minimum distance and size feature, additionally to the medulla dummy feature. Therefore, this cluster largely positively correlates to minimum distance, size and medulla. Similarly, the right cluster positively correlates with the maximum, centre and weighted distance, additionally to the parotid gland features. On the other hand, the same three features are on the opposite side of the PC1 to the left cluster. Thus, the left cluster, which positively correlates to the submandibular glands, negatively correlates with the maximum, centre and weighted distance.

The explained variance graph in Figure 4.19 on page 73 illustrates that seven PCs are needed for explaining almost 100 % of the variance to the test set. However, only five PCs are needed to explain 90 % of the variance. Together, PC1 and PC2 explain 57 % (36.2 % + 20.8 %). Thus, 57 % of the total variance of the test set is explained by the first two PCs.

Further, there is about a 15 % increase of the explained variance when adding PC3 and PC4. Therefore, the explained variance plot suggests that it might be interesting to analyse a PCA for PC3 and PC4 also. Although there are no PCA plots using the PC3 and PC4 illustrated in this thesis, the correlation loadings plot with PC3 and PC4 suggests that the two PCs explain the variance between the left and right parotid and submandibular glands. No variance of the medulla or any spatial features contributed to the variance in PC3 or PC4.

The coloured scores plot in Figure 4.20 on page 73 demonstrate whether the associated instances were correctly or incorrectly predicted by the Logistic Regression. Figure 4.20 show that all three clusters have many incorrectly predicted instances. Moreover, there does not seem to be a pattern between the correctly or incorrectly

instances. Thus, the type of OAR is not responsible for the inaccuracy of the model. Additionally, coloured scores plot using other PCs gave similar results with no systematic patterns.

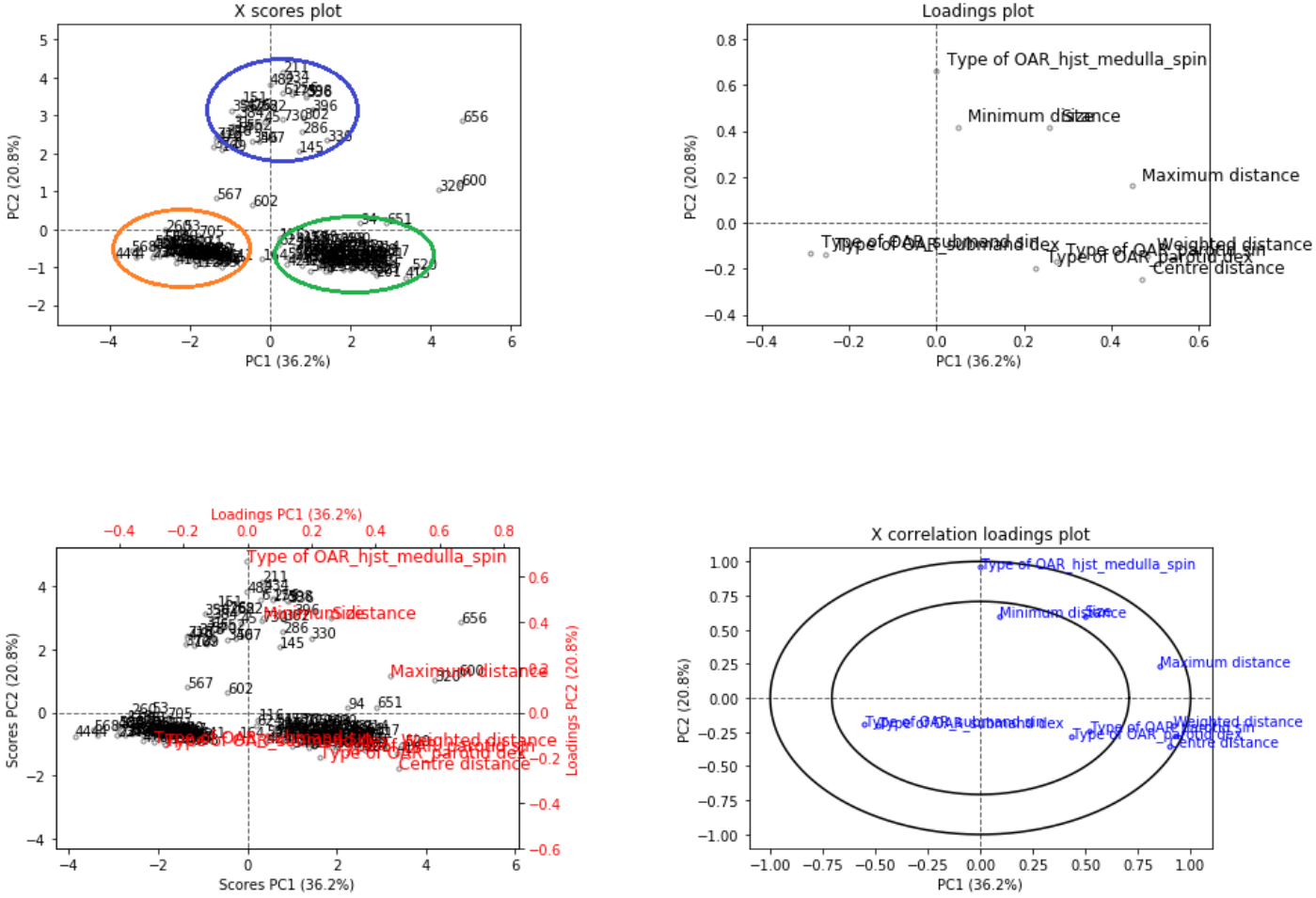



Figure 4.18: Scores plot, loadings plot, biplot and correlation loadings plot for test set using the first two PCs. The associated explained variance is noted beside each PC. The labels attached to the loadings are the associated feature names, and the scores' names are random index number from the feature matrix. The circles of clustered scores in the scores plot reflect the correlation each score cluster has with the OAR type, blue = medulla, orange = submandibular glands, green = parotid glands.

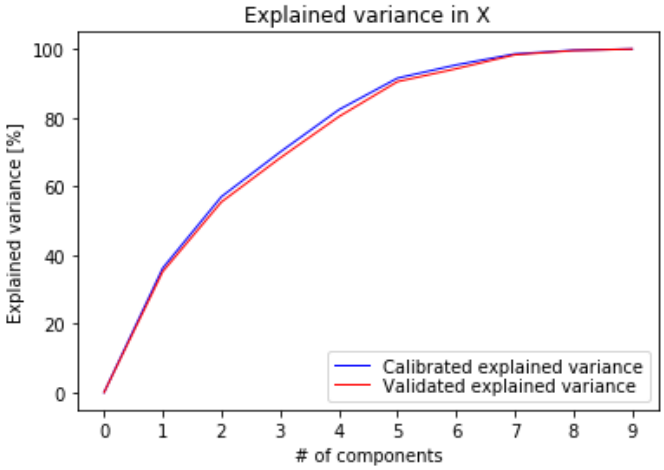



Figure 4.19: Explained variance plot for the test set used in classification. The blue line represents the calibrated explained variance, and the red line represents explained variance validated with CV.

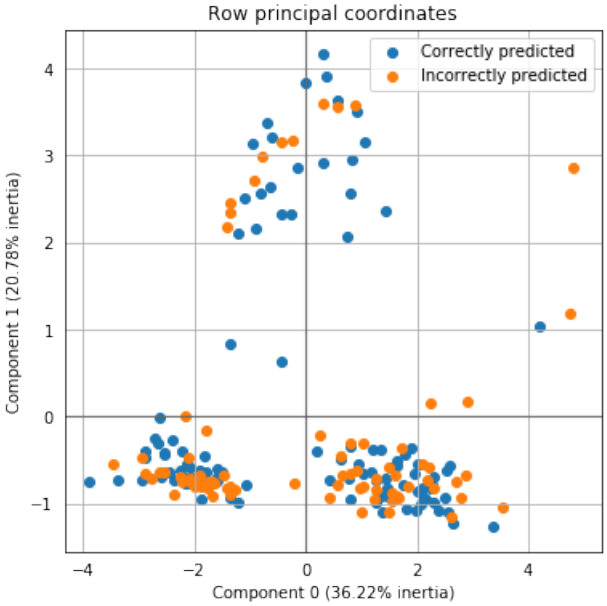



Figure 4.20: Coloured scores plot for test set used in classification for the first two PCs. The correctly predicted instances are coloured in blue, and the incorrectly predicted instances are orange. Component 0 and 1 correspond to PC1 and PC2 from Figure 4.18. Inertia refers to the calibrated explained variance.

### 4.2.3 Regression

#### Initial regression

Predicting the dose using the first Random Forest regressor resulted in the $R^2$ scores of the training and test sets to be 0.951 and 0.695, respectively. Thus, the Random Forest model fit quite well with the data. However, the model suffered from overfitting, which is also reflected in the large difference in value with the *RMSE* scores of 3.844 (train) and 9.316 (test).

Figure 4.21 is a distribution plot of the absolute difference between the true and predicted mean doses of the test set. In the distribution plot, most values are less than 18 Gy, and a large portion have less than 8 Gy difference. Additionally, the distribution is reflected in its statistical measurements of median = 5, mean = 7 and standard deviation = 6.

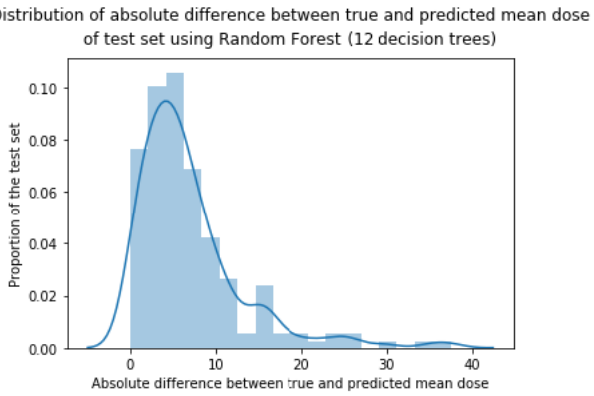



Figure 4.21: Distribution plot of the absolute difference between true and predicted mean dose for the test set using the Random Forest with 12 decision trees.

#### Feature selection using Sequential Backward Floating Selection

Figure 4.22 on page 76 and Figure 4.23 on page 77 are illustrations of the Sequential Backward Floating Selection (SBFS) processes. Both SBFS processes eliminated the size of OARs first, which increased the average $R^2$ score. Thus, the SBFS indicates that the size feature should not be included for predicting the mean dose.

The final features of the SBFS with `merge_glands = False` in Figure 4.22 were maximum distance, centre distance and brain stem / medulla. This differed from the results from the SBFS in Figure 4.13. However, as illustrated in Figure 4.18 on page 72, the medulla and minimum distance seem to correlate to a certain degree. Therefore, the SBFS processes agree that the information contributed by the minimum distance or brain stem is relevant data, yet disagree on which feature to use.

On the other hand, the SBFS with `merge_glands = True` in Figure 4.23 concur with the SBFS in Figure 4.14 on page 65. The dummy feature for the submandibular glands is the feature which alone can give the highest score of the Random Forest. The reason might be that the submandibular glands often receive a large mean dose, as explained with Figure 4.8 on page 59. Thus, if the instance is a submandibular salivary gland, the algorithm could achieve a high score with predicting a large dose for all submandibular salivary glands.

The parotid glands' dummy feature seems to contribute information by inspecting Figure 4.22, but not by reviewing Figure 4.23. Therefore, it is difficult to be certain about the contribution of this feature.

### Regressor comparisons

The highest $R^2$ score achieved from either SBFS processes was with the feature maximum distance, centre distance, weighted distance, brain stem / medulla, parotid and submandibular, with 0.789 as $R^2$ value. Different regressors were evaluated using this feature combination. Additionally, the feature combination containing only maximum distance, centre distance, and weighted distance was also evaluated. The results are listed in Table 4.2 on page 78.

The Random Forest algorithm with quadratic polynomial features gave the highest $R^2$ score. Nonetheless, the Random Forest with no quadratic polynomial features only differed with 0.01. Moreover, the feature combination containing the dummy OAR features achieved a higher score for all regressors.

### Hyperparameter tuning and evaluation of highest performing regressor

The grid search for the tuned Random Forest with quadratic polynomial features resulted in 40 decision trees, the *MSE* as the quality measure function, and the logarithm to the base of 2 for calculating the number of features to use for a split.

The Random Forest with the above hyperparameters, and using quadratic polynomial features, resulted in a $R^2$ score of 0.969 and 0.647 for the training and test set, respectively. The *RMSE* values were 3.067 and 10.029. Consequently, the tuned Random Forest was more overfitted than for the first Random Forest regressor. Additionally, the tuned Random Forest achieved a lower score for the test set.

Without the quadratic polynomial features, the *RMSE* values of the training and test sets were 2.945 and 9.962. Hence, the polynomial features did not necessarily contribute to more information for predicting the mean dose received. The $R^2$ scores were 0.972 (train) and 0.651 (test).

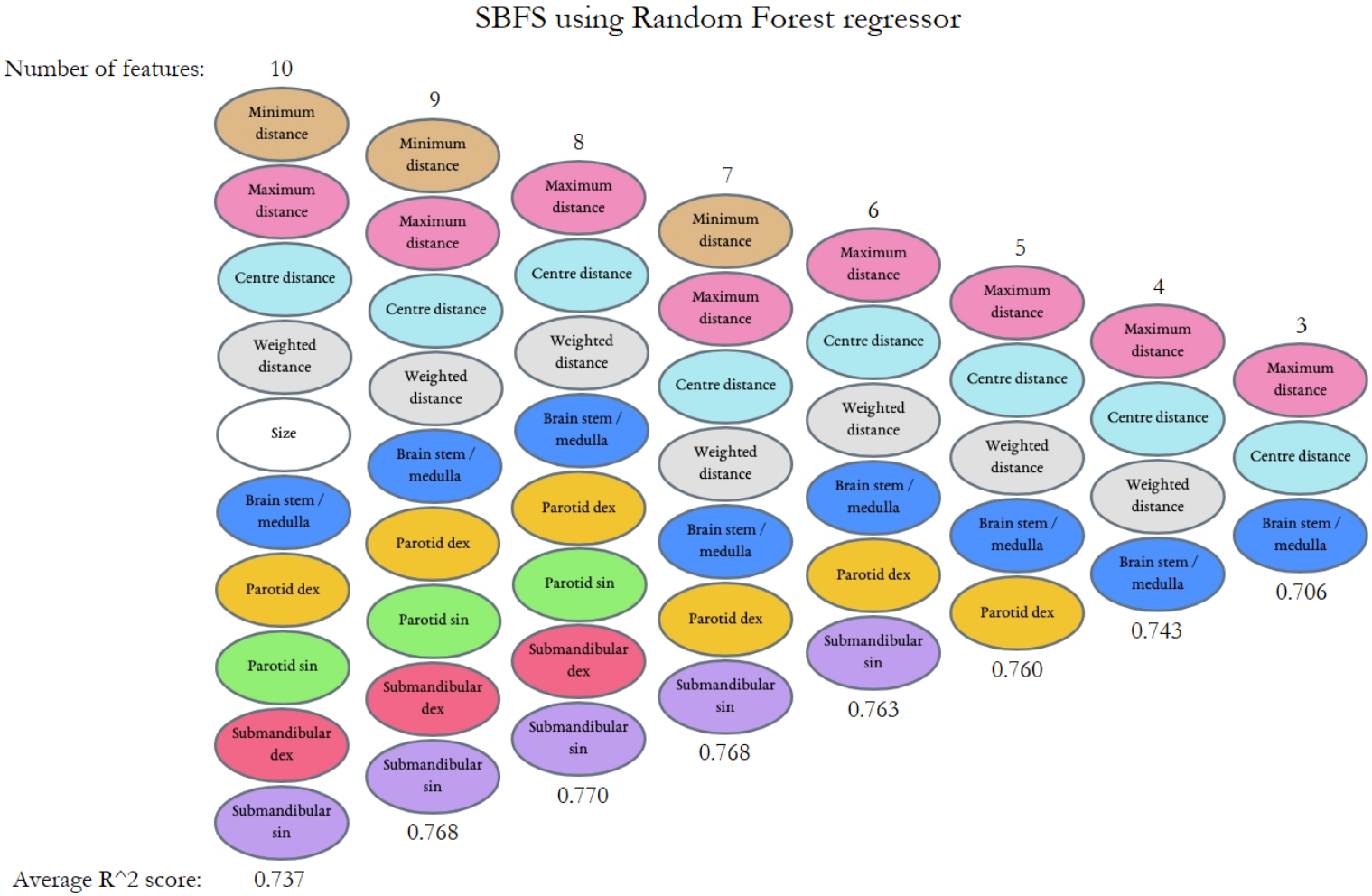



Figure 4.22: The SBFS result for regression when `merge_glands = False`. The far-left column represents the first iteration where the SBFS has used all features available. The far-right column illustrates the three features the SBFS decided to keep. The columns from left to right represents a feature being removed for each iteration step of the SBFS. Additionally, the features are depicted with a specific colour. The number of features are written above each iteration column, and the $R^2$ score is noted below.

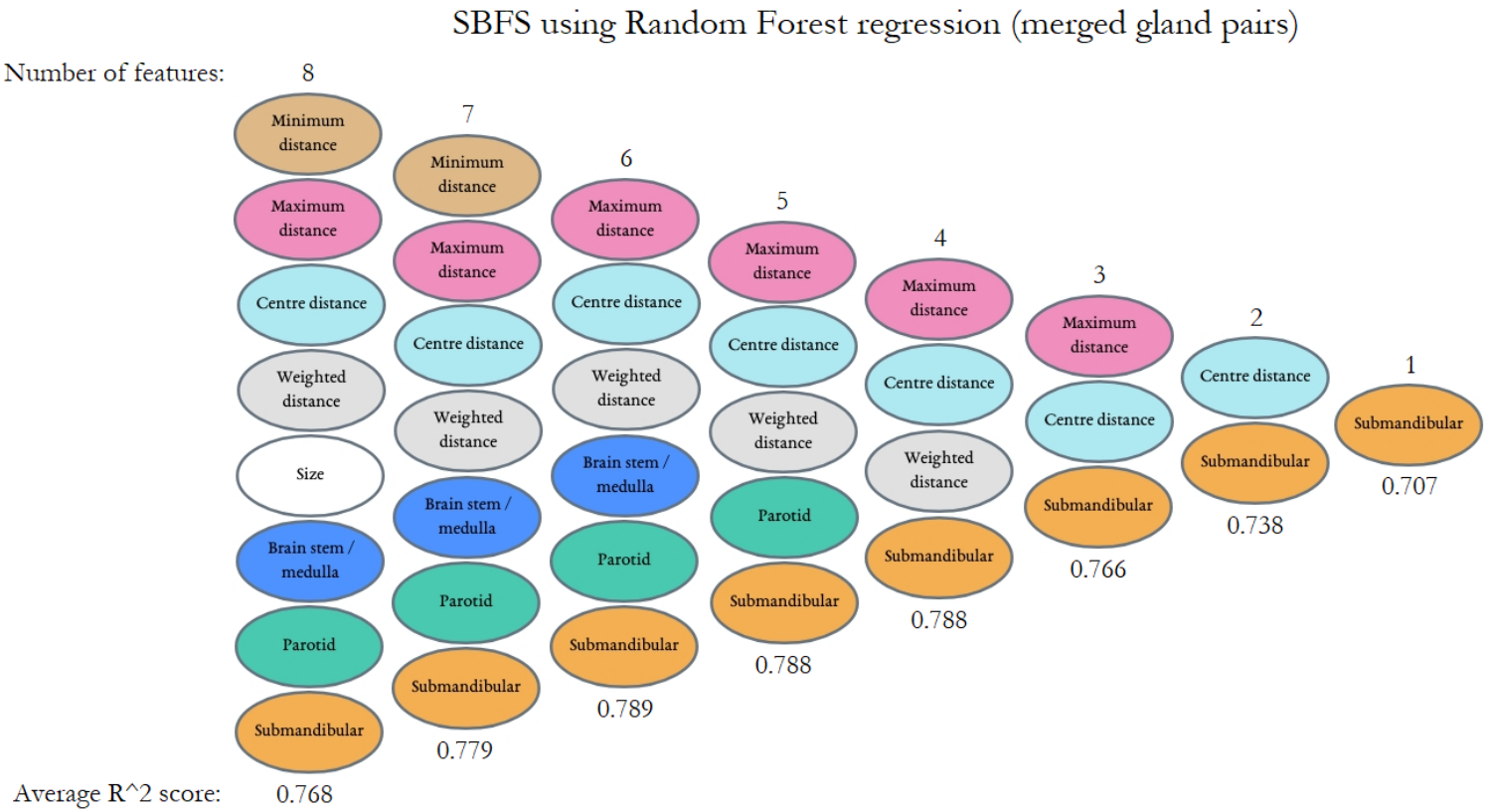



Figure 4.23: The SBFS result for regression when `merge_glands = False`. The far-left column represents the first iteration where the SBFS has used all features available. The far-right column illustrates the one feature the SBFS decided to keep. The columns from left to right represents a feature being removed for each iteration step of the SBFS. Additionally, the features are depicted with a specific colour. The number of features are written above each iteration column, and the $R^2$ score is noted below.

Table 4.2: Result of cross validating different regressors with the average $R^2$ score (+/- standard deviation of the CV scores). The only instance containing merged pair glands is the feature combination of Maximum distance, Centre distance, Weighted distance, Brain stem / medulla, Parotid and Submandibular.

| | **Random Forest** | **Decision Trees** | **Random Forest with Quadratic Polynomial features** | **LASSO** | **SVR** | **AdaBoost** |
|---|---|---|---|---|---|---|
| Maximum distance, Centre distance, Weighted distance, Brain stem / medulla, Parotid, Submandibular | 0.77 (+/- 0.075) | 0.66 (+/- 0.11) | 0.78 (+/- 0.075) | 0.74 (+/- 0.07) | 0.76 (+/- 0.071) | 0.74 (+/- 0.081) |
| Maximum distance, Centre distance, Weighted distance | 0.65 (+/- 0.13) | 0.36 (+/- 0.2) | 0.66 (+/- 0.078) | 0.48 (+/- 0.1) | 0.57 (+/- 0.094) | 0.6 (+/- 0.072) |

# Chapter 5

# Discussion

The aim for this thesis was to map dose distributions of target PTVs and OARs for both photon and proton therapy. Further, spatial statistic features were extracted from CT images and dose plans to predict doses given to OARs. By predicting dose received from different radiotherapy approaches, a machine learning model could assist radiologists by addressing cancer patients' treatment needs.

Both photon and proton dose plans were supposed to be compared after OAR dose prediction was devised. However, the proton data was not available yet. Nevertheless, the exploration and analysis done could be applied to a similar proton data set. Therefore, the focus of the thesis shifted to solely predicting dose received by OARs using photon dose plans.

The machine learning models constructed to predict the mean dose of OARs suffered from underfitting. The complexity of the models should therefore be increased. Due to the fact that no model combination of features or learners resulted in a high score with few prediction errors, it is likely that the features did not provide sufficient explanation of the dose distributions.

In spite of the learning models' inaccuracies, the exploratory data analysis results showed that some of the extracted features correlated strongly with mean dose of the OARs. Thus, the indication is that the correlating features explained the dose of the OARs to some degree. Therefore, these features, or other variants of them, could be used for predicting mean dose to an OAR.

## 5.1 Feature relations to the mean OAR dose

The strong linear correlations between dose received and centre distance, weighted distance and maximum distance between OARs and PTV found in the exploratory

data analysis implied that these features could be important for a machine learning model. However, these distance features correlate strongly with each other as well. Therefore, these features might explain the same dose variation of the OARs.

The distance between OARs and PTV was the characteristic that was initially assumed to highly correlate with the OARs received dose [14]. The dose plan will ensure a high dose to the PTV, regardless of an OAR's distance to the PTV. The exploratory data analysis showed that the centre distance correlated strongly with the mean dose. Accordingly, the iterations of the Sequential Backward Floating Selection (SBFS) processes deemed the centre distance an important feature for the learning models. Therefore, the centre distance could be a feature suited for explaining the connection between OAR distance to the PTV and its dose.

Although maximum distance correlate with centre and weighted distance, the feature could be contributing to separate information regarding the PTV and OARs' spatial relations. For the two SBFS processes that were instructed to finish with only one feature (illustrated with Figure 4.14 and Figure 4.23), both resulted in a decreased score when maximum distance was removed. A decrease in score also occurred when weighted distance was removed during the SBFS processes in Figure 4.22 and Figure 4.23. Thus, the maximum and weighted distance might contribute to relevant information, because they describe the spatial relations between PTV and OAR, which may also affect the mean dose to an OAR.

On the other hand, the reason for the weighted distance's high correlation with mean and median dose might be because of the high correlation with centre distance, and not due to additional information. Also, all SBFS processes resulted in the centre distance to be prioritised higher than the weighted distance.

Moreover, the weighted distance was constructed on the premise of the bones' attenuation coefficient. The bones receive a higher dose than soft tissue [36]. Thus, radiation penetrates soft tissue easier than bones. However, the photoelectric effect is responsible for the elevated dose to bones [71]. Yet, for x-rays used in radiotherapy, the contribution of the photoelectric effect is relatively small [72]. Therefore, the difference in attenuation coefficients between bones and soft tissue do not necessarily affect the radiation received by OARs during radiotherapy of HNC.

There was a very weak correlation found between the spatial feature minimum distance and mean dose in the correlation matrix Figure 4.1 on page 52. However, the overlap volume, which relates to the minimum distance, correlated moderately with the mean dose for each type of OAR (illustrated in Figure 4.5 and Figure 4.6 on page 56). On the other hand, the overlap Dice, which relates to the features minimum distance, overlap volume and size of structures, had a smaller linear correlation with mean dose to OAR. By this, it is likely that combining size with overlap volume adds noise to the data. This hypothesis is supported by the interpretation of the SBFS procedures, which deemed the size feature insignificant.

The size of the OARs should in theory relate to the type of OAR, although no analysis was done to support this hypothesis. Nonetheless, it would be a possible option as to why the size feature moderately correlated to the mean dose, as shown with the correlation matrix in Figure 4.1 on page 52. Yet, the size feature was rejected by the SBFS processes for learning. The dummy features representing type of OARs were kept in the SBFS procedures, suggesting that they were significant. Moreover, if the type of OAR reflects size, the dummies might be the desired features to explain this variance without too much fluctuations. The interaction between dose distributions and type of OARs are demonstrated in both Figure 4.4 on page 55 and Figure 4.8 on page 59. However, the relationship of size and OAR type should be analysed before concluding.

Although the machine learning models achieved the highest scores using the type of OAR features, the goal should be to not use these. A model could become biased towards OAR type, and prioritise OAR type higher than spatial circumstances. As observed with the dose distributions in Figure 4.8 on page 59, submandibular glands received a larger average dose than parotid glands. Nonetheless, some parotid glands received large mean doses as well. Furthermore, most medullas received a low dose. However, for a model to be reliable, the instances which deviates from the norm should be detected. Thus, a learning model should not be biased towards OAR type.

## 5.2 Strengths and weaknesses of the machine learning procedure

The advantage of implementing a regression method over a classification method is that the mean dose feature, which manages continuous variables, can be used directly as target labels for a regression model. When the continuous variables are transformed into classes, there is an information loss of the mean dose variance. The information loss is observed in the scores of the different learning methods. The difference between true and predicted class labels only describes the score of the model using a 12 Gy range per category. On the other hand, regression models, which employ the continuous variables directly, describe the difference explicitly for each unit of Gy. Therefore, regression may give a more accurate representation of scores when the regression metrics rely on continuous variables.

Despite the advantage of regression over classification for this data set, neither model would have been reliable to use for predicting mean dose to OARs. The Random Forest tested initially for classification clearly overfitted, whereas the tuned Logistic Regression underfitted. Thus, even though the Logistic Regression predicted about half of the instances correctly, the score is too low for the model to be a reliable predictor.

Similarly, the tuned Random Forest for regression was not deemed reliable either. Tuning the Random Forest regressor resulted in overfitting. Adjusting the number of decision trees is the hyperparameter which generally has the largest effect on the fit [41]. Therefore, the increase in overfitting from the first Random Forest to the tuned Random Forest was most likely due to increasing the number of decision trees by a factor of four.

An attempted approach for increasing the feature space was done by transforming the data set with polynomial regression of degree two [41]. However, adding the quadratic features to the data set before the Random Forest regressor might not have been necessary, because Random Forest is a non-linear learning algorithm. Although, if the Random Forest finds the new features unnecessary, the algorithm would try to disregard these when splitting the data. On the other hand, if the new features increased the information of variance, the algorithm might split the data differently, which can lead to a higher prediction accuracy.

To gain more accurate scores of the algorithms implemented, the process of tuning, training and testing should have been done for several different data partitions. Also, feature selection should have been done for different combinations of training and validation sets and other algorithms. Thus, the repeated processes could be used to gain a more accurate score of the algorithm. Additionally, a feature evaluation technique, such as the drop-out feature importance process [48], could have been implemented to confirm the contribution each feature would have had on the score. However, it is unlikely that the scores of the models would drastically improve by these processes.

A solution for improving model scores could be to increase the data set size. A machine learner can always learn from more data, and the underfitting of the classification model implies a lack of information. The 225 patients in the data set for this thesis is not necessarily a small data set. Nonetheless, it is small enough that one needs to consider whether the data splits were representative enough. The training data needs to be representative of the data to build a robust model, but the test data also needs to be representative to make accurate predictions. Yet, it is difficult to achieve both when the data set is small.

However, the data size is not of primary importance if the features cannot be used for constructing an accurate prediction model. The underfitting of the classification model, and the test scores of the regression model, may imply that the features extracted are not sufficient to construct a robust dose prediction model. Although the features explain some aspects of mean dose given to an OAR, there might be other factors than the features extracted that affect the dose. Thus, these other factors should be included in the dose prediction model.

## 5.3 Other approaches

A couple of recent studies (Nguyen et al. [73] and (Chen et al. [74]) used convolutional neural networks (CNNs) to predict the radiotherapy dose distributions for prostate and nasopharyngeal cancer patients. In short, CNN is a type of deep learning, where deep learning is a branch of machine learning that uses a specific layer paradigm for learning [41]. CNNs specialise in images, and use image recognition to find patterns and thus features from images [75].

Nguyen et al. [73] stated that their CNN extracted features based on shape, size and location of the PTVs and OARs, and thus learned to predict dose using this data. Furthermore, with an average Dice coefficient of 0.91 between the true and predicted dose distribution volumes, the network accurately predicted the dose.

A CNN could be a machine learning option that could be tested on the HNC data set. The features extracted in this thesis could not accurately predict the dose. Also, creating and extracting new features could take time, and may not result in accurate predictions. Thus, a CNN could both extract and predict the dose of a patient. Moreover, the method has been shown to give accurate results for prostate and nasopharyngeal cancer patients. Therefore, it may also work on the HNC data set.

Other uses of machine learning within oncology is often related to predicting potential after-effects of radiotherapy [76][77][78][79]. Outcome predictions are important when creating treatment plans for patients to minimise harm and painful after-effects, such as xerostomia for HNC. However, these learning algorithms are not directly linked to evaluating photon versus proton therapy, or to the prioritisation process of patients for proton therapy. Nonetheless, there are other model types implemented that are consistently improving for evaluating radiotherapy treatments.

A type of model implemented within oncology is the Normal Tissue Complication Probability (NTCP) [80]. The NTCP is an estimation of probability that a given radiation dose will cause damage to an organ. Therefore, the NTCP is a useful tool for treatment planning when evaluating different treatment plans. Although the equation of a NTCP is the same, factors affecting the probability for individual OARs differ depending on circumstances [8][9][10].

NTCP models have been applied to the use of evaluating photon versus proton therapy, and also for the prioritisation process [81][82][83][84][85]. Similarly as to comparing treatment plans when predicting outcomes for OARs, the NTCP models can be used for illustrating the difference between photon and proton plans. Thus, the objective of comparing treatment plans for the prioritisation process is the same, whether using a NTCP or a machine learning model. However, the NTCP model is already an established evaluation method, whereas more research is required before

using machine learning for similar predictions [77][78][79].

## 5.4 Future work

With the aim of predicting and comparing photon and proton plans for patient prioritisation, the proton dose plans of the same patients used in this thesis should also be processed and examined. Thus, hypotheses of similarities and dissimilarities of the different treatment plans could then be established.

Furthermore, a patient selection model of proton therapy should include other factors than spatial features. External factors, such as age, also influence the decision of treatment method [6]. A machine learning model can be trained using relevant external factors in addition to the dose plans.

Furthermore, a treatment prioritisation algorithm could be trained using assessments of radiologists as response variables. The assessments would contain the results of radiologists' evaluation of patients' treatment plan needs. Therefore, such assessments must be done before training a learning algorithm for patient selection.

The features extracted in this thesis did not result in reliable models. Therefore, future work should involve obtaining other features that can predict mean dose to OARs accurately. Nguyen et al. [73] mentioned that their CNN managed to predict both PTVs' and OARs' dose distributions based on the sizes, locations and shapes of the structures. Therefore, the findings of Nguyen et al. suggest that there are spatial circumstances that can accurately predict dose distributions, yet the same features were most likely not used in this thesis.

Another suggestion could be to test a similar CNN architecture to the one introduced by Nguyen et al. or Chen et al. for the HNC data set. If a CNN would be developed for predicting the dose received by OARs, then one does not need to predetermine features to extract.

A disadvantage of using deep learning for dose prediction or patient selection is that it would become more difficult to explain the features. Since the algorithm finds its features from analysing the data set on its own, the algorithm can find patterns which may not be logical in the real world. Thus, the features are not necessarily easy to explain, which they should be if they are used for the important task of deciding treatment plans of cancer patients in the future.

# Chapter 6

# Conclusion

Machine learning could be a useful tool for prioritising radiotherapy treatment of HNC patients. The dose predictions of OARs for both proton and photon radiotherapy could give insight into a patient's treatment needs. The spatial features extracted from the HNC data correlated with the mean dose, however, they did not result in accurate or robust prediction models. Nevertheless, groups of researchers with a similar goal of predicting dose distributions for prostate and nasopharyngeal cancer patients achieved highly accurate models by using a different machine learning approach. Therefore, further research of features and other machine learning techniques should be tested for dose predictions and radiotherapy treatment prioritisation.

# Bibliography


[1] Cancer Research UK, *Worldwide cancer statistics*, Accessed: 2019-02-27. [Online]. Available: https://www.cancerresearchuk.org/health-professional/cancer-statistics/worldwide-cancer.

[2] Fred Hutchinson Cancer Research Center, *A Message from Dr. Gary Gilliland about cancer research in 2018*, Accessed: 2019-01-15. [Online]. Available: https://www.fredhutch.org/en/news/cancer-research-2018.html.

[3] Gianfaldoni S., Gianfaldoni R., Wollina U., Lotti J., Tchernev G., Lotti T., 'An Overview on Radiotherapy: From Its History to Its Current Applications in Dermatology', *Open Access Macedonian Journal of Medical Sciences*, vol. 5, pp. 45–53, 2017. DOI: 10.3889/oamjms.2017.122.

[4] National Cancer Institute, *Types of Cancer Treatment*, Accessed: 2019-01-15. [Online]. Available: https://www.cancer.gov/about-cancer/treatment/types.

[5] Particle Therapy Co-Operative Group, *Particle therapy facilities in clinical operation (last update: February 2019)*, Accessed: 2019-02-27. [Online]. Available: https://www.ptcog.ch/index.php/facilities-in-operation.

[6] Dale E., Waldeland E., *Protonterapi – en realitet i norge fra 2023*, Accessed: 2019-01-10. [Online]. Available: https://tidsskriftet.no/2018/09/kronikk/protonterapi-en-realitet-i-norge-fra-2023.

[7] Kalveland J., *Få pasienter henvises til protonterapi i utlandet*, Accessed: 2019-04-19. [Online]. Available: https://www.dagensmedisin.no/artikler/2019/01/24/fa-pasienter-henvises-til-protonterapi-i-utlandet/.

[8] Beetz I., Schilstra C., Burlage F. R., Koken P. W., Doornaert P., Bijl H. P., Chouvalova O., Leemans C. R., de Bock G. H., Christianen M. E., van der Laan B. F., Vissink A., Steenbakkers R. J., Langendijk J.A., 'Development of NTCP models for head and neck cancer patients treated with three-dimensional conformal radiotherapy for xerostomia and sticky saliva: The role of dosimetric and clinical factors', *Radiotherapy and Oncology*, vol. 105, pp. 86 –93, 2012. DOI: 10.1016/j.radonc.2011.05.010.

[9] Christianen M. E., Schilstra C., Beetz I., Muijs C. T., Chouvalova O., Burlage F. R., Doornaert P., Koken P. W., Leemans C. R., Rinkel R. N., de Bruijn M. J., de Bock G. H., Roodenburg J. L., van der Laan B. F., Slotman B. J., Verdonck-de Leeuw I. M., Bijl H. P., Langendijk J. A., 'Predictive modelling for swallowing dysfunction after primary (chemo)radiation: Results of a prospective observational study', *Radiotherapy and Oncology*, vol. 105, pp. 107 –114, 2012. DOI: 10.1016/j.radonc.2011.08.009.

[10] Wopken K., Bijl H. P., van der Schaaf A., van der Laan H. P., Chouvalova O., Steenbakkers R. J., Doornaert P., Slotman B. J., Oosting S. F., Christianen M. E., van der Laan B. F., Roodenburg J. L., Leemans C. R., Verdonck-de Leeuw I. M., Langendijk J. A., 'Development of a multivariable normal tissue complication probability (NTCP) model for tube feeding dependence after curative radiotherapy/ chemo-radiotherapy in head and neck cancer', *Radiotherapy and Oncology*, vol. 113, pp. 95 –101, 2014. DOI: 10.1016/j.radonc.2014.09.013.

[11] The National Association for Proton Therapy, *How Does Proton Therapy Work?*, Accessed: 2019-01-15. [Online]. Available: https://www.proton-therapy.org/science/.

[12] Blanchard P., Gunn G. B., Lin A., Foote R. L., Lee N. Y., Frank S. J., 'Proton Therapy for Head and Neck Cancers', *Seminars in Radiation Oncology*, vol. 28, pp. 53 –63, 2018.

[13] The University of Texas MD Anderson Cancer Center, *Proton Therapy for Childhood Cancers*, Accessed: 2019-01-22. [Online]. Available: https://www.mdanderson.org/patients-family/diagnosis-treatment/care-centers-clinics/proton-therapy-center/conditions-we-treat/pediatric-cancer.html.

[14] *Discussion between radiologists and medical physicists at the Bioradiance meeting, Radiohospitalet. 17-12-2018*, Personal communication.

[15] Leeman J. E., Romesser P. B., Zhou Y., McBride S., Riaz N., Sherman E., Cohen M. A., Cahlon O., Lee N., 'Proton therapy for head and neck cancer: Expanding the therapeutic window', *The Lancet Oncology*, vol. 18, pp. 254 –256, 2017. DOI: 10.1016/S1470-2045(17)30179-1.

[16] SAS Institute Inc., *Machine Learning: What it is and why it matters*, Accessed: 2019-01-15. [Online]. Available: https://www.sas.com/en_us/insights/analytics/machine-learning.html.

[17] DataRobot, *Feature Variables*, Accessed: 2019-01-15. [Online]. Available: https://www.datarobot.com/wiki/feature/.

[18] Mayo Clinic, *Proton Therapy*, Accessed: 2019-01-29. [Online]. Available: https://www.mayoclinic.org/tests-procedures/proton-therapy/about/pac-20384758.

[19] American Cancer Society, *External Beam Radiation Therapy*, Accessed: 2019-05-12. [Online]. Available: https://www.cancer.org/treatment/treatments-and-side-effects/treatment-types/radiation/external-beam-radiation-therapy.html.

[20] NDT Resource Center, *X-Radiation*, Accessed: 2019-04-25. [Online]. Available: https://www.nde-ed.org/EducationResources/CommunityCollege/Radiography/Physics/xrays.htm.

[21] The University of Texas MD Anderson Cancer Center, *History of Proton Therapy*, Accessed: 2019-01-23. [Online]. Available: https://www.mdanderson.org/patients-family/diagnosis-treatment/care-centers-clinics/proton-therapy-center/what-is-proton-therapy/history-of-proton-therapy.html.

[22] ProTom - Proton Therapy Technologies, *The Evolution of Proton Therapy: Accelerating Cancer Treatment*, Accessed: 2019-01-23. [Online]. Available: https://www.protominternational.com/2018/05/history-of-proton-therapy/.

[23] Peeters A., Grutters J. P., Pijls-Johannesma M., Reimoser S., De Ruysscher D., Severens J. L., Joore M. A., Lambin P., 'How costly is particle therapy? Cost analysis of external beam radiotherapy with carbon-ions, protons and photons', *Radiotherapy and Oncology*, vol. 95, pp. 45 –53, 2010.

[24] Bordvik M., – *ikke verdt pengene*, Accessed: 2019-02-08. [Online]. Available: https://www.dagensmedisin.no/artikler/2016/08/03/-ikke-verdt-pengene/.

[25] Orenstein B. W., 'Proton Therapy and Cost', *Radiology Today*, vol. 16, p. 22, 2015. [Online]. Available: https://www.radiologytoday.net/archive/rt0215p22.shtml.

[26] Durante M., Orecchia R., Loeffler J. S., 'Charged-particle therapy in cancer: Clinical uses and future perspectives', *Clinical Oncology*, vol. 14, pp. 483 –495, 2017.

[27] Prayongrat A., Umegaki K., van der Schaaf A., Koong A. C., Lin S. H., Whitaker T., McNutt T., Matsufuji N., Graves E., Mizuta M., Ogawa K., Date H., Moriwaki K., Ito YM., Kobashi K., Dekura Y., Shimizu S., Shirato H., 'Present developments in reaching an international consensus for a model-based approach to particle beam therapy', *Journal of Radiation Research*, vol. 59, pp. 72 –76, 2018. DOI: 10.1093/jrr/rry008.

[28] Berthelsen A. K., Dobbs J., Kjellén E., Landberg T., Möller T. R., Nilsson P., Specht L., Wambersieg A., 'What's new in target volume definition for radiologists in ICRU Report 71? How can the ICRU volume definitions be integrated in clinical practice?', *Cancer Imaging*, vol. 7, pp. 104 –116, 2007. DOI: 10.1102/1470-7330.2007.0013.

[29] Leer J. W. H., 'What the clinician wants to know: Radiation oncology perspective', *Cancer imaging : the official publication of the International Cancer Imaging Society*, vol. 5, S1 –S2, 2005. DOI: 10.1102/1470-7330.2005.0027.

[30] Brain Made Simple, *Medulla Oblongata*, Accessed: 2019-02-14. [Online]. Available: http://brainmadesimple.com/medulla-oblongata.html.

[31] *Communication with Eirik Malinen concerning the dataset and considerations to make when creating the program*, Personal communication.

[32] Kreftforeningen, *CT*, Accessed: 2019-03-11. [Online]. Available: https://kreftforeningen.no/om-kreft/undersokelse-ved-kreft/ct/.

[33] NDT Resource Center, *Transmitted Intensity and Linear Attenuation Coefficient*, Accessed: 2019-03-11. [Online]. Available: https://www.nde-ed.org/EducationResources/CommunityCollege/Radiography/Physics/attenuationCoef.htm.

[34] Goldman L. W., 'Principles of CT and CT Technology', *Journal of Nuclear Medicine Technology*, vol. 35, pp. 115 –128, 2007. DOI: 10.2967/jnmt.107.042978.

[35] Mayfield Brain & Spine, *Computed tomography (CT) and CT angiography*, Accessed: 2019-04-05. [Online]. Available: https://mayfieldclinic.com/pe-ct.htm.

[36] Case Western Reserve University, *Computed Tomography (CT) of the Brain: Basics*, Accessed: 2019-04-05. [Online]. Available: http://casemed.case.edu/clerkships/neurology/Web%20Neurorad/CT%20Basics.htm.

[37] Lev M. H., Gonzalez R. G., in *Brain Mapping: The Methods)*, Second Edition, San Diego: Academic Press, 2002, pp. 427 –484, ISBN: 978-0-12-693019-1. DOI: 10.1016/B978-012693019-1/50019-8.

[38] Bibb R, Eggbeer D, Paterson A., in *Medical Modelling*, Second Edition, Oxford: Woodhead Publishing, 2015, pp. 7 –34, ISBN: 978-1-78242-300-3. DOI: 10.1016/B978-1-78242-300-3.00002-0.

[39] Xue Z., Antani S., Long L. R., Demner-Fushman D, Thoma G. R., 'Window Classification of Brain CT Images in Biomedical Articles', *American Medical Informatics Association*, vol. 2012, pp. 1023–1029, 2012.

[40] Universitetssykehuset Nord-Norge, *Strålebehandling, Informasjonshefte for pasienter*, Accessed: 2019-03-11. [Online]. Available: https://unn.no/seksjon-avdeling/Documents/Avdelinger/Kreftavdelingen/Str%C3%A5leterapi_130315.pdf.

[41] Raschka S., Mirjalili V., *Python Machine Learning*, 2nd ed. Birmingham, UK: Packt Publishing Ltd., 2017, ISBN: 978-1-78712-593-3.

[42] McKinney W., 'Data Structures for Statistical Computing in Python', in *Proceedings of the 9th Python in Science Conference*, M. J. van der Walt S., Ed., 2010, pp. 51 –56.

[43] Shah T., *About Train, Validation and Test Sets in Machine Learning*, Accessed: 2019-04-29. [Online]. Available: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7.

[44] Brownlee J., *How Much Training Data is Required for Machine Learning?*, Accessed: 2019-04-29. [Online]. Available: https://machinelearningmastery.com/much-training-data-required-machine-learning/.

[45] Tomic O., *DAT200 - Applied machine learning: Data preprocessing*, Lecture: 2018-03-15, Ås: Norges miljø- og biovienskapelige universitet (NMBU).

[46] Pedregosa F., Varoquaux G., Gramfort A., Michel V. Thirion B., Grisel O., and Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E., 'Scikit-learn: Machine Learning in Python', *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[47] Brownlee J., *Overfitting and Underfitting With Machine Learning Algorithmst*, Accessed: 2019-04-19. [Online]. Available: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/.

[48] Tomic O., *DAT300 - Applied machine learning II: Feature selection and evaluation*, Lecture: September 2018, Ås: Norges miljø- og biovienskapelige universitet (NMBU).

[49] McDonald C., *Machine learning fundamentals (I): Cost functions and gradient descent*, Accessed: 2019-04-19. [Online]. Available: https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220.

[50] Shung K. P., *Accuracy, Precision, Recall or F1?*, Accessed: 2019-05-07. [Online]. Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

[51] Drakos G., *How to select the right evaluation metric for machine learning models: Part 1 regression metrics*, Accessed: 2019-04-22. [Online]. Available: https://towardsdatascience.com/how-to-select-the-right-evaluation-metric-for-machine-learning-models-part-1-regrression-metrics-3606e25beae0.

[52] Tomic O., *DAT200 - Applied machine learning: Introduction to machine learning*, Lecture: 2018-02-12, Ås: Norges miljø- og biovienskapelige universitet (NMBU).

[53] ——, *DAT200 - Applied machine learning: Basic machine learning, classification*, Lecture: 2018-02-19, Ås: Norges miljø- og biovienskapelige universitet (NMBU).

[54] James G., Witten D., Hastie T., Tibshirami R., *An Introduction to Statistical Learning*. New York: Springer, 2013, ISBN: 978-1-4614-7138-7.

[55] Koehrsen W., *Hyperparameter Tuning the Random Forest in Python*, Accessed: 2019-04-29. [Online]. Available: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74.

[56] Hatt M., Lee J. A., Schmidtlein C. R., Naqa I. E., Caldwell C., De Bernardi E., Lu W., Das S., Geets X., Gregoire V., Jeraj R., MacManus M. P., Mawlawi O. R., Nestle U., Pugachev A. B., Schöder H., Shepherd T., Spezi E., Visvikis D., Zaidi H., Kirov A. S., 'Classification and evaluation strategies of auto-segmentation approaches for PET: Report of AAPM task group No. 211.', *Medical Physics*, vol. 44, e1 –e42, 2017. DOI: 10.1002/mp.12124.

[57] Neuromorphometrics, Inc., *Dice Similarity Coefficients (DSCs), How Good is "Good Enough"?*, Accessed: 2019-03-21. [Online]. Available: http://www.neuromorphometrics.com/?p=457.

[58] Shahid R., Bertazzon S., Knudtson M. L., Ghali W. A., 'Comparison of distance measures in spatial analytical modeling for health service planning', *BMC Health Services Research*, vol. 9, 2009. DOI: 10.1186/1472-6963-9-200.

[59] College of Saint Benedict and Saint John's University, *Box plot: Display of distribution*, Accessed: 2019-02-26. [Online]. Available: http://www.physics.csbsju.edu/stats/box2.html.

[60] Statistics Solutions, *Correlation (Pearson, Kendall, Spearman)*, Accessed: 2019-03-12. [Online]. Available: https://www.statisticssolutions.com/correlation-pearson-kendall-spearman/.

[61] statstutor, *Spearman's correlation*, Accessed: 2019-03-12. [Online]. Available: http://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf.

[62] WolframMathWorld, *Monotonic function*, Accessed: 2019-03-20. [Online]. Available: http://mathworld.wolfram.com/MonotonicFunction.html.

[63] Powell V., Lehe L., *Principal Component Analysis*, Accessed: 2019-04-15. [Online]. Available: http://setosa.io/ev/principal-component-analysis/.

[64] Campbell N. A., Atchley W. R., 'The geometry of canonical variate analysis', *Systematic Zoology*, vol. 30, no. 3, pp. 268–280, 1981. DOI: 10.2307/2413249.

[65] Tomic O., *DAT200 - Applied machine learning: Subspace analysis*, Lecture: 2018-02-05, Ås: Norges miljø- og biovienskapelige universitet (NMBU).

[66] Kvaal K., *INF250 - 9 Teori Prosjekt*, Lecture 2016-12-05, Ås: Norges miljø- og biovienskapelige universitet (NMBU).

[67] Dunn K., *Principal Component Analysis (PCA)*, Accessed: 2019-04-15, 2019. [Online]. Available: https://learnche.org/pid/latent-variable-modelling/principal-component-analysis/index.

[68] Unkel S., *Principal component analysis (PCA):Principles, Biplots, and Modern Extensions forSparse Data*, Accessed: 2019-04-16, University Medical Center Göttingen, 2017. [Online]. Available: http://www.ams.med.uni-goettingen.de/download/Steffen-Unkel/chap6.pdf.

[69] Swalin A., *Choosing the Right Metric for Evaluating Machine Learning Models — Part 1*, Accessed: 2019-05-03. [Online]. Available: https://medium.com/usf-msds/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4.

[70] Donges N., *The Random Forest Algorithm*, Accessed: 2019-05-01. [Online]. Available: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd.

[71] Bazalova M., Graves E. E., 'The importance of tissue segmentation for dose calculations for kilovoltage radiation therapy', *Medical Physics*, vol. 38, pp. 3039 –3049, 2011. DOI: 10.1118/1.3589138.

[72] Flint P. W., Haughey B. H, Robbins K. T, Thomas J. R., Niparko J. K., Lund V. J., Lesperance M. M., *Cummings Otolaryngology - Head and Neck Surgery*, 6th ed. Philadelphia, PA, U.S.A.: Saunders Elsevier, 2015, ISBN: 978-1-4557-4696-5.

[73] Nguyen D., Long T., Jia X., Lu W., Gu X., Iqbal Z., Jiang S., 'A feasibility study for predicting optimal radiation therapy dose distributions of prostate cancer patients from patient anatomy using deep learning', *Scientific Reports*, vol. 9, 2019. DOI: 10.1038/s41598-018-37741-x.

[74] Chen X, Men K., Li Y., Yi J., Dai J., 'A feasibility study on an automated method to generate patient-specific dose distributions for radiotherapy using deep learning', *Medical Physics*, vol. 46, pp. 56 –64, 2019. DOI: 10.1002/mp.13262.

[75] Rouse M., *Convolutional Neural Network*, Accessed: 2019-05-07. [Online]. Available: https://searchenterpriseai.techtarget.com/definition/convolutional-neural-network.

[76] Dean J. A., Wong K. H., Welsh L. C., Jones A. B., Schick U., Newbold K. L., Bhide S. A., Harrington K. J., Nutting C.M., Gulliford S. L., 'Normal tissue complication probability (NTCP) modelling using spatial dose metrics and machine learning methods for severe acute oral mucositis resulting from head and neck radiotherapy', *Radiotherapy and Oncology*, vol. 120, pp. 21 –27, 2016.

[77] Gabryś H.S., Buettner F., Sterzing F., Hauswald H., Bangert M., 'Design and Selection of Machine Learning Methods Using Radiomics and Dosiomics for Normal Tissue Complication Probability Modeling of Xerostomia', *Frontiers in Oncology*, vol. 8, 2018.

[78] Lee T. F., Liou M. H., Huang Y. J., Chao P. J., Ting H M., Lee H. Y., Fang F. M., 'LASSO NTCP predictors for the incidence of xerostomia in patients with head and neck squamous cell carcinoma and nasopharyngeal carcinoma', *Scientific Reports*, vol. 4, 2014. DOI: 10.1038/srep06217.

[79] Soares I., Dias J., Rocha1 H., Khouri L., Lopes M. C., Ferreira B., 'Predicting xerostomia after IMRT treatments: a data mining approach', *Health and Technology*, vol. 8, pp. 159 –168, 2018. DOI: 10.1007/s12553-017-0204-4.

[80] Michalski D., Huq M. S., Hasson, B. F., 'Normal Tissue Complication Probability (NTCP)', in *Encyclopedia of Radiation Oncology*, Brady L. W., Yaeger T. E., Ed. Berlin, Heidelberg: Springer, 2013, pp. 560 –560, ISBN: 978-3-540-85516-3.

[81] Prayongrat A., Umegaki K., van der Schaaf A., Koong A. C., Lin S. H., Whitaker T., McNutt T., Matsufuji N., Graves E., Mizuta M., Ogawa K., Date H., Moriwaki K., Ito Y. M., Kobashi K., Dekura Y., Shimizu S., Shirato H., 'Present developments in reaching an international consensus for a model-based approach to particle beam therapy', *Journal of Radiation Research*, vol. 59, pp. i72 –i76, 2018. DOI: 10.1093/jrr/rry008.

[82] Arts T., Breedveld S., de Jong M. A., Astreinidou E., Tans L., Keskin-Cambay F., Krol A. D. G., van de Water S., Bijman R. G., Hoogeman M. S., 'The impact of treatment accuracy on proton therapy patient selection for oropharyngeal cancer patients', *Radiotherapy and Oncology*, vol. 123, pp. 520 –525, 2017.

[83] Kobashi K., Prayongrat A., Kimoto T., Toramatsu C., Dekura Y., Katoh N., Shimizu S., Ito Y. M., Shirato H., 'Assessing the uncertainty in a normal tissue complication probability difference (?NTCP): radiation-induced liver disease (RILD) in liver tumour patients treated with proton vs X-ray therapy', *Journal of Radiation Research*, vol. 59, pp. i50 –i57, 2018. DOI: 10.1093/jrr/rry018.

[84] Ramaekers B. L., Grutters J. P., Pijls-Johannesma M., Lambin P., Joore M. A., Langendijk J. A., 'Protons in Head-and-Neck Cancer: Bridging the Gap of Evidence', *International Journal of Radiation Oncology, Biology, Physics*, vol. 85, pp. 1282 –1288, 2013.

[85] Kierkels R. G. J., Fredriksson A., Both S., Langendijk J. A., Scandurra D., Korevaar E. W., 'Automated Robust Proton Planning Using Dose-Volume Histogram-Based Mimicking of the Photon Reference Dose and Reducing Organ at Risk Dose Optimization', *International Journal of Radiation Oncology, Biology, Physics*, vol. 103, pp. 251 –258, 2019.

# Appendix

Table 6.1: Table of Python IDEs.

| IDE | Version |
|---|---|
| Jupyter Notebook | 5.7.8 |
| Pycharm | 2018.3.5 |
| Spyder | 3.3.3 |

Table 6.2: Table of Python packages. If there is no version listed for a package, then this is a module of Python, and will correspond to the version of Python used.

| Package | Version |
|---|---|
| collections | |
| numpy | 1.16.2 |
| math | |
| matplotlib | 3.0.3 |
| mlxtend | 0.15.0.0 |
| pandas | 0.24.2 |
| pathlib | |
| pickle | |
| prince | 0.6.2 |
| scipy | 1.2.1 |
| seaborn | 0.9.0 |
| skimage | 0.14.2 |
| sklearn | 0.20.3 |
| statistics | |

# create_features.ipynb

## 1 Functions for creating features

Create features of structures:

- Size of PTVs and OARs
- Mean and median dose to PTVs and OARs
- Distance between PTVs and OARs using their centre of masses
- Maximum and minimum distance between PTVs and OARs
- Weighted distance using CT values

Also calculates overlap volume and overlap dice if structures have a minimum distance of zero. All features determined are set into dictionaries and saved as pickle- files.

### 1.1 Imports

```
In [1]: import numpy as np
        from pathlib import Path
        import pickle

        from scipy.ndimage.measurements import center_of_mass
        from scipy.spatial.distance import cdist, euclidean
        from statistics import median
        from skimage.morphology import erosion
```

### 1.2 Size of structures

```
In [2]: def get_sizes(num_patients, struct_keys):
            """
            Creates and saves dictionary with the sizes of OARs and PTVs for all
            patients up to the number given.

            Parameters
            ----------
            num_patients : int
                Number of patients.

            struct_keys : dict
                The numbered keys for the different structures.
```

```
    Returns
    -------
        None
    """

    nono = [20, 45, 70, 72, 121]

    for nr in range(1, num_patients + 1):
        if nr in nono:
            pass
        else:
            nr = str(nr)
            ptv_file = Path('D:\\Protonpro\\pas_' + nr + '\\ptv.npy')
            org_file = Path('D:\\Protonpro\\pas_' + nr + '\\organs.npy')

            if org_file.is_file():
                org = np.load(org_file)

            else:
                org = np.array([0])

            if ptv_file.is_file():
                ptv = np.load(ptv_file)

            else:
                ptv = np.array([0])

            org_and_ptv = [list(org), list(ptv)]

            numbers = [list(np.unique(nu)) for nu in org_and_ptv]
            numbers = [item for sublist in numbers for item in sublist]
            numbers = list(set(numbers))
            numbers.remove(0)

            sizes = {}

            for uniquenr in numbers:
                if uniquenr in org:
                    struct = org
                elif uniquenr in ptv:
                    struct = ptv
                else:
                    raise ValueError('numbers list is incorrect')

                li = np.argwhere(struct == uniquenr)
                size = len(li)
                for name, nu in struct_keys.items():
```

```
                    if nu == uniquenr:
                        sizes[name] = size

            with open('D:\\Protonpro\\pas_' + nr + '\\sizes.pkl', 'wb') as file:
                pickle.dump(sizes, file, pickle.HIGHEST_PROTOCOL)
```

### 1.3 Mean and median dose to PTVs and OARs

```
In [3]: def calculate_mean_median_dose_for_volume(coordinates_for_object, dose_plan):
            """
            Calculates the mean and median dose for the object which has the given
            coordiantes.

            Parameters
            ----------
            coordinates_for_object : ndarray
                2 dimensional ndarray where the zyx-coordinates are expressed in each
                column, meaning each row represents a coordinate where there is
                identified an object.

            dose_plan : ndarray
                The dose plan for the associated patient.

            Returns
            -------
                int, int
                The mean and median dose of the object.
            """

            tot_dose_obj = 0
            doses = []
            for coor in coordinates_for_object:
                dose_vox = dose_plan[tuple(coor)]
                tot_dose_obj += dose_vox
                doses.append(dose_vox)

            if tot_dose_obj > 0:
                rel_tot_dose_obj = tot_dose_obj / len(coordinates_for_object)
            else:
                rel_tot_dose_obj = tot_dose_obj

            return int(round(rel_tot_dose_obj)), int(round(median(doses)))

In [4]: def mean_median_dose_per_object(num_patients, struct_keys):
            """
            Saves median and mean doses for all OARs and PTV for each patient.

            Parameters
```

```
    ----------
    num_patients : int
        The number of patients there are.

    struct_keys : dict
        The numbered keys for the different structures.

    Returns
    -------
        None
    """

    nono = [20, 45, 70, 72, 121]

    for nr in range(1, num_patients + 1):
        if nr in nono:
            pass
        else:

            ptv_file = Path('D:\\Protonpro\\pas_' + str(nr) + '\\ptv.npy')
            org_file = Path('D:\\Protonpro\\pas_' + str(nr) + '\\organs.npy')
            dose = np.load('D:\\Protonpro\\pas_' + str(nr) + '\\dose.npy')

            mean_doses_per_volume = {}
            median_doses_per_volume = {}

            if org_file.is_file():

                org = np.load(org_file)

                uniq_org = list(np.unique(org))
                uniq_org.remove(0)

                for obj_nr in uniq_org:
                    obj = np.argwhere(org == obj_nr)

                    mean, medi = calculate_mean_median_dose_for_volume(obj,
                                                                       dose)

                    obj_name = str(
                        (list(struct_keys.keys())[
                            list(struct_keys.values()).index(obj_nr)]))

                    mean_doses_per_volume[obj_name] = mean
                    median_doses_per_volume[obj_name] = medi

            if ptv_file.is_file():
```

```
                ptv = np.load(ptv_file)

                uniq_ptv = list(np.unique(ptv))
                uniq_ptv.remove(0)

                for obj_nr in uniq_ptv:
                    obj = np.argwhere(ptv == obj_nr)

                    mean, medi = calculate_mean_median_dose_for_volume(obj,
                                                                       dose)

                    obj_name = str(
                        (list(struct_keys.keys())[
                            list(struct_keys.values()).index(obj_nr)]))

                    mean_doses_per_volume[obj_name] = mean
                    median_doses_per_volume[obj_name] = medi

            with open('D:\\Protonpro\\pas_' + str(nr) + '\\mean_doses.pkl',
                      'wb') as file:
                pickle.dump(mean_doses_per_volume, file,
                            pickle.HIGHEST_PROTOCOL)

            with open('D:\\Protonpro\\pas_' + str(nr) + '\\median_doses.pkl',
                      'wb') as file:
                pickle.dump(median_doses_per_volume, file,
                            pickle.HIGHEST_PROTOCOL)
```

## 1.4 Distance between PTVs and OARs using their centre of masses, and weighted distance using CT values

```
In [5]: def find_centres_org_ptv(struct1, struct2, struct_keys):
            """
            Finds the centre of masses for all labels of the two three-dimensional
            images given.
            All points have been rounded to 1 decimal.

            Parameters
            ----------
            struct1 : ndarray
                The first structures to find the centre of masses of.

            struct2 : ndarray
                The second structures to find the centre of masses of.

            struct_keys : dict
                The numbered keys for the different structures.
```

```
        Returns
        -------
        dict
            Centre of mass points for the first structures.

        dict
            Center of mass points for the second structures.
        """

        nrs_1 = list(np.unique(list(struct1)))
        nrs_1.remove(0)
        nrs_2 = list(np.unique(list(struct2)))
        nrs_2.remove(0)

        centre1 = center_of_mass(struct1, struct1, nrs_1)
        centre2 = center_of_mass(struct2, struct2, nrs_2)

        for i, centre in enumerate(centre1):
            centre1[i] = (
                int(round(centre[0])), int(round(centre[1])), int(round(centre[2])))

        for i, centre in enumerate(centre2):
            centre2[i] = (
                int(round(centre[0])), int(round(centre[1])), int(round(centre[2])))

        struct1_centres = {}
        struct2_centres = {}

        for i, nr in enumerate(nrs_1):
            struct = str(
                (list(struct_keys.keys())[list(struct_keys.values()).index(nr)]))
            struct1_centres[struct] = centre1[i]

        for i, nr in enumerate(nrs_2):
            struct = str(
                (list(struct_keys.keys())[list(struct_keys.values()).index(nr)]))
            struct2_centres[struct] = centre2[i]

        return struct1_centres, struct2_centres

In [6]: def line_points(p1, p2):
        """
        Finds certain number of points on the line between the coordinates p1 and
        p2 in 3D.

        Parameters
        ----------
        p1 : tuple
```

```
        Coordinates for the first point.

    p2 : tuple
        Coordinates for the second point.

    Returns
    -------
    list
        Coordinates given as tuples that are on the line between p1 and p2.
    """

    diff = [abs(p2[0] - p1[0]), abs(p2[1] - p1[1]), abs(p2[2] - p1[2])]

    nb_points = max(diff)

    x_spacing = (p2[0] - p1[0]) / (nb_points + 1)
    y_spacing = (p2[1] - p1[1]) / (nb_points + 1)
    z_spacing = (p2[2] - p1[2]) / (nb_points + 1)

    return ([(round(p1[0] + i * x_spacing), round(p1[1] + i * y_spacing),
              round(p1[2] + i * z_spacing)) for i in range(1, nb_points + 1)])

In [7]: def get_centre_and_ct_distances(num_patients, struct_keys):
    """
    Compute and save the distances and distance*sum(CT values) between OARs'
    and PTVs' center of mass points.

    Parameters
    ----------
    num_patients : int
        The number of patients there is data on.

    struct_keys : dict
        Overview of which OARs/PTVs respond to which integer in an ndarray.

    Returns
    -------
        None
    """

    nono = [20, 45, 70, 72, 121]

    for nr in range(1, num_patients + 1):
        if nr in nono:
            pass
        else:

            ptv_file = Path('D:\\Protonpro\\pas_' + str(nr) + '\\ptv.npy')
```

```
                org_file = Path('D:\\Protonpro\\pas_' + str(nr) + '\\organs.npy')

                if org_file.is_file() and ptv_file.is_file():
                    org = np.load(org_file)
                    ptv = np.load(ptv_file)
                    ct = np.load(Path('D:\\Protonpro\\pas_' + str(nr) + '\\ct.npy'))

                    centre_org, centre_ptv = find_centres_org_ptv(org, ptv,
                                                                  struct_keys)

                    distances = {}
                    dist_with_ct = {}
                    for organ in list(centre_org.keys()):
                        for tumor in list(centre_ptv.keys()):
                            dist = int(round(euclidean(
                                centre_org[organ], centre_ptv[tumor])))

                            points = line_points(centre_org[organ],
                                                 centre_ptv[tumor])
                            ct_values = [ct[poi] for poi in points]

                            distances[organ + ' to ' + tumor] = dist
                            dist_with_ct[organ + ' to ' + tumor] = int(
                                round(dist * sum(ct_values)))

                    with open('D:\\Protonpro\\pas_' + str(
                            nr) + '\\centre_distances.pkl',
                              'wb') as file:
                        pickle.dump(distances, file, pickle.HIGHEST_PROTOCOL)

                    with open(
                            'D:\\Protonpro\\pas_' + str(nr) + '\\ct_distances.pkl',
                            'wb') as file:
                        pickle.dump(dist_with_ct, file, pickle.HIGHEST_PROTOCOL)
```

### 1.5 Maximum and minimum distance between PTVs and OARs

```
In [8]: def get_contours(structure):
            """
            Creates the outline of structures in a ndarray.

            Parameters
            ----------
            structure : ndarray
                Structures to get the outline of.

            Returns
            -------
```

```
            ndarray
                Outline of structures.
            """

            return structure - erosion(structure)

In [9]: def compute_min_max_dist(a1, a2):
            """
            Computes the minimum and maximum Euclidean distances between two arrays.

            Parameters
            ----------
            a1 : array
                First array.
            a2 : array
                Second array.

            Returns
            -------
            tuple
                Minimum and maximum distance between a1 and a2.
            """
            distances = cdist(a1, a2)

            return (int(round(distances.min())), int(round(distances.max())))

In [10]: def save_min_max_distances(num_patients, struct_keys):
             """
             Calculates and saves the minimum and maximum distances between PTV and OAR
             structures for all patients. Also, for the OARs that have a minimum
             distance to a PTV of 0, the overlap volume and dice is determined and
             saved in separate pickle- files.

             Parameters
             ----------
             num_patients : int
                 The number of patients there are.

             struct_keys : dict
                 The numbered keys for the different structures.

             Returns
             -------
                 None
             """

             nono = [20, 45, 70, 72, 121]
```

```
for nr in range(1, num_patients + 1):
    if nr in nono:
        pass
    else:

        ptv_file = Path('D:\\Protonpro\\pas_' + str(nr) + '\\ptv.npy')
        org_file = Path('D:\\Protonpro\\pas_' + str(nr) + '\\organs.npy')

        if org_file.is_file() and ptv_file.is_file():

            org = np.load(org_file)
            ptv = np.load(ptv_file)

            cont_org = get_contours(org)
            cont_ptv = get_contours(ptv)

            uniq_org = list(np.unique(org))
            uniq_org.remove(0)
            uniq_ptv = list(np.unique(ptv))
            uniq_ptv.remove(0)

            min0_nr_overlap = {}
            min0_dice = {}
            distances = {}
            for org_nr in uniq_org:
                org_nr_cont = np.argwhere(cont_org == org_nr)
                org_name = str(
                    (list(struct_keys.keys())[
                        list(struct_keys.values()).index(org_nr)]))

                for ptv_nr in uniq_ptv:
                    ptv_nr_cont = np.argwhere(cont_ptv == ptv_nr)
                    ptv_name = str(
                        (list(struct_keys.keys())[
                            list(struct_keys.values()).index(ptv_nr)]))

                    minandmax = compute_min_max_dist(
                        org_nr_cont, ptv_nr_cont)
                    distances[org_name + ' to ' + ptv_name] = minandmax

                    if minandmax[0] == 0:
                        this_ptv = np.argwhere(ptv == ptv_nr)
                        this_oar = np.argwhere(org == org_nr)

                        toar = list(map(tuple, this_oar))
                        tptv = list(map(tuple, this_ptv))

                        nr_toar = len(toar)
```

```
                        nr_tptv = len(tptv)

                        overlap = list(set(tptv).intersection(set(toar)))
                        nr_overlap = len(overlap)
                        min0_nr_overlap[
                            org_name + ' to ' + ptv_name] = nr_overlap
                        min0_dice[
                            org_name + ' to ' + ptv_name] =\
                            nr_overlap * 2 / (nr_toar + nr_tptv)

                with open('D:\\Protonpro\\pas_' + str(
                        nr) + '\\min_max_distances.pkl',
                          'wb') as file:
                    pickle.dump(distances, file, pickle.HIGHEST_PROTOCOL)

                if min0_nr_overlap:
                    with open('D:\\Protonpro\\pas_' + str(
                            nr) + '\\min0_nr_overlap.pkl',
                              'wb') as file:
                        pickle.dump(min0_nr_overlap, file,
                                    pickle.HIGHEST_PROTOCOL)

                if min0_dice:
                    with open(
                            'D:\\Protonpro\\pas_' + str(nr) + '\\min0_dice.pkl',
                            'wb') as file:
                        pickle.dump(min0_dice, file, pickle.HIGHEST_PROTOCOL)
```

### 1.6 Code to start creating features

```
In [ ]: with open('D:\\Protonpro\\keys.pkl',
                  'rb') as file:
            keys = pickle.load(file)

        num = 230

        get_sizes(num, keys)
        mean_median_dose_per_object(num, keys)
        get_centre_and_ct_distances(num, keys)
        save_min_max_distances(num, keys)
```

**Norges miljø- og biovitenskapelige universitet**
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway