Norwegian University
of Life Sciences

Master's Thesis 2021   30 ECTS
Faculty of Science and Technology

Predicting treatment outcome of colorectal cancer from MRI images using machine learning

Marthe Susann Søvdsnes
Data Science

Acknowledgements

This thesis marks the end of my studies in Data science at the Norwegian University of Life Sciences NMBU. I would like to thank my supervisors for all the support and patience. I would also like to thanks my parents for helping me start this journey in the first place and continued support throughout all my years at NMBU.

As for my friends, I would like to thank all for the good times over the years and especially my flatmates for keeping life interesting during 2020 and 2021.

Abstract

In this thesis different machine learning algorithms have been utilised to predict treatment outcome for patients with colorectal cancer. The predicted treatment endpoint was overall survival. The patient cohort included 77 patients with histologically confirmed colorectal cancer who were recruited at Akershus University Hospital between 2013 and 2017. Radiomics was used to extract first-order statistics, shape and texture features from T2-weighed images and DWIs taken of the patients before starting treatment. These features, in addition to clinical data, were used to train machine learning models. The models were later combined into majority vote classifiers. Models and majority vote classifiers were built for three different patient subsets: all patients, patients who had received chemoradiotherapy, and patients who had not received chemoradiotherapy. Performance was estimated using k-fold cross validation with MCC as the the validation metric.

Repeated Elastic Net Technique (RENT) and PCA were used for feature reduction before training models and building the majority vote classifiers. RENT was also used to analyse feature importance for the radiomics data.

All the majority vote classifiers achieved mean MCC scores above 0, but had quite large mean standard deviations. The differences in the performances of the models between folds in the k-fold cross validation were severe, indicating that the data was susceptible to poor train-test splits. A handful of features with high selection frequency were singled out during the RENT analysis of the radiomics data.

Contents

1 Introduction **8**

2 Theory **10**

2.1 Cancer 10

2.1.1 Colorectal Cancer 11

2.1.2 Staging 11

2.1.3 Treatment 12

2.2 Magnetic resonance imaging 12

2.2.1 Fundamental physics for MRI 13

2.2.2 T1 and T2 weighed images 14

2.2.3 Diffused weighed images 15

2.3 Machine learning 16

2.3.1 Classification 16

2.3.2 Preprocessing 17

2.3.2.1 Standardisation 17

2.3.2.2 Challenges with high-dimensional data 17
2.3.2.3 Repeated Elastic Net Technique 18
2.3.2.4 Principal component analysis 19
2.3.3 Training process 20
2.3.3.1 Train and test set 20
2.3.3.2 Cross validation 21
2.3.3.3 Over- and Underfitting 22
2.3.3.4 Class imbalance 23
2.3.3.5 Regularisation 23
2.3.3.6 Hyperparameter Tuning 24
2.3.4 Classification algorithms 25
2.3.4.1 Support Vector Machine 25
2.3.4.2 Logistic Regression 26
2.3.4.3 Decision Trees 27
2.3.4.4 Random forest 28
2.3.4.5 K-nearest neighbours 29
2.3.5 Ensemble learning 29
2.3.6 Evaluation 30
2.3.6.1 Confusion matrix 30
2.3.6.2 Simple evaluation statistics 30
2.3.6.3 Matthews correlation coefficient 31

3 Materials and methods **32**

3.1 Radiomics . . . 33

3.1.1 First-order statistics features . . . 33

3.1.2 Shape features . . . 34

3.1.3 Texture features . . . 34

3.2 MRI data files . . . 34

3.3 Data selection . . . 35

3.4 Splitting the datasets . . . 38

3.5 Workflow . . . 39

3.6 Validation . . . 41

3.7 Software . . . 43

4 Results **44**

4.1 Results from all-patients subsets . . . 45

4.2 Results from CRT subsets . . . 49

4.3 Results from no-CRT subsets . . . 53

4.4 Selected features . . . 56

4.4.1 Clinical data features . . . 56

4.4.2 Radiomics features . . . 58

5 Discussion **64**

5.1 Interpreting results . . . 64

5.2 Evaluation process 65

5.3 Selected features 66

5.4 Robustness of radiomic features 67

5.5 Selected models and data sets 67

5.6 Choice of response variable 68

5.7 Parameter choices 69

5.8 Personalised medicine 70

5.9 Future work 70

6 Conclusion **72**

Bibliography **75**

A Models **76**

A.1 Models from the all-patients subsets 76

A.2 Models from the CRT subsets 79

A.3 Models from the no-CRT subsets 82

B Python code **85**

B.1 Machine learning algorithms 85

B.2 Data preprocessing 89

B.3 Feature reduction 95

B.4 Workflow 98

Chapter 1

Introduction

Cancer is one of the leading causes of death in the developed world [1]. In 2020 alone, as many as 10.0 million deaths were attributed to cancer and 19.3 million new cases were reported worldwide [2]. Magnetic resonance imaging (MRI) is used when diagnosing cancer. MRIs give valuable information about a patients prognosis and play a factor in deciding the correct course of treatment [3].

Radiomics is a growing field in medicine. It is based on the hypothesis that MRI and images from other modalities, such as PET and CT scans, contain valuable information that is not available through visual inspection alone. Radiomics is the process of extracting quantitative data from these images via mathematical algorithms [4].

The goal of radiomics is to give a better understanding of how different tumours respond to treatments, as well as which patients are more at risk for certain factors, such as increased risk of complications or cancer recurrence. A deeper understanding of such subjects can greatly influence the cancer treatment options that are given to individual patients. This can allow dosing, treatment duration and intensity to be tailored for each patient.

This thesis has two aims. The first aim is to predict treatment outcome for patients with colorectal cancer with overall survival as the endpoint. This will be done by applying machine learning methods on information extracted from MRIs and available clinical data. The second aim is to analyse which, if any, features extracted from MRIs might hold medical predictive power.

To achieve the first aim, a number of majority vote classifiers were constructed. The classifiers were made in the hope of increasing predictive power by combining models based on different datasets. The classifiers were made up of three different models trained on radiomics and clinical data. The reasoning behind using majority vote classifiers is that each dataset might have strengths and weaknesses when it comes to detecting and predicting certain elements in the data. Some elements might be more prominent in one set than another, and a model from one set could pick up on different aspects of the data compared to other models.

To achieve the second aim, a number of Repeated Elastic Net Technique (RENT) analyses were performed. RENT is a feature selection technique that focuses on feature selection stability.

Chapter 2

Theory

2.1 Cancer

Cancer is a catch-all term for a group of about 200 diseases that involve abnormal cell growth. The body is constantly producing new cells to replace old ones. New cells are created when a cell doubles its DNA and splits in two. Sometimes a mutation occurs in a new cell that causes it to split itself uncontrollably and the mutated cells will begin to form a tumour. The time between the mutation occurring and the tumour becoming detectable varies based on cancer type, group and aggressiveness. Cancer can spread to other organs by way of the bloodstream or lymphatic system. This is called metastasis [5].

Cancer kills by affecting major organs and their ability to function [5] and is one of the leading causes of death in the developed world [1]. In 2020 alone, 10.0 million deaths were attributed to cancer and 19.3 million new cases were reported worldwide [2].

2.1.1 Colorectal Cancer

Colorectal cancer refers to cancer originating in the large bowel (colon) and the back passage (rectum) [6]. It is the most common type of cancer in Norway and affects men and women equally. 4.499 new cases of colorectal cancer were reported in Norway in 2019, 2.338 men and 2.161 women [7]. The cause is usually unknown, but according to [5] and [8] increased risk has been linked to a number of factors. Some of these factors are common across all cancer types and some factors are specific to colorectal cancer.

Common factors include diets with an excess of red and processed meat and lacking in fibers, being overweight or obese, lack of physical activity, drinking alcohol or smoking, old age, and exposure to radiation, among others. Specific factors for colorectal cancer include some inherited conditions, such as familial adenomatous polyposis and Lynch syndrome, other medical conditions such as Ulcerative colitis and Crohn's disease, as well as the number of benign polyps present in the bowel.

2.1.2 Staging

Cancer staging is the process of finding out how far a cancer has progressed. Staging is done before starting treatment and is important in deciding which treatment a patients receives. Staging is based on where the tumour is located, the size of the tumour and whether the cancer has spread to nearby lymph nodes or other parts of the body. There are many different staging systems, but the TNM system is the most widely used [9].

In the TNM system, T describes the primary tumour, N describes whether the cancer has spread to nearby lymph nodes and how many, and M describes whether the cancer has spread to other parts of the body. Each letter is followed by a number or the letter X. The number describes the extent of the cancer relating to the specific area covered by the letter. If a letter is followed by X it means that the cancer could not be measured.

T followed by 0 means that abnormal cells have been found but have not yet formed a tumour. T followed by 1 or higher describes the extent of the main tumour. The higher the number the larger the tumor is in size or the more the tumour has grown into nearby tissues. The highest T staging is 4.

N followed by 0 means that the cancer has not spread to any lymph nodes. N followed by 1 or higher means that the cancer has spread to nearby lymph nodes. A high number means that more lymph nodes contain cancer compared to a lower number. The highest N staging is 3.

M followed by 0 means that the cancer has not metastasised. If M is followed by a 1 it means that the cancer has spread. This is the highest M staging [9].

2.1.3 Treatment

Treatment of colorectal cancer depends on factors such as the stage of the cancer, the placement of the tumour and the general condition of the patient [10]. The main treatment options for cancer are surgery, radiotherapy, chemotherapy and combined chemotherapy and radiotherapy, also know as chemoradiotherapy (CRT) [11].

Radiation therapy works by damaging the DNA of cells in order to kill or stop cancer cells from dividing [12]. Chemotherapy refers to drugs used for cancer treatment. Radiation and surgery are local treatments, meaning that they are aimed at specific parts of the body where cancer has been found. Chemotherapy on the other hand travels throughout the body and can therefore kill cancer cells that have metastasised [13].

Patients might receive CRT before surgery if the cancer has spread to nearby structures or tissue, or to shrink the tumour to create clear tissue borders for surgery [11].

2.2 Magnetic resonance imaging

Magnetic resonance imaging (MRI) is an imaging modality that uses non-ionising radiation to create diagnostic images. An MRI scanner consists of a powerful magnet in which the patient lies and radio frequency (RF) transmit and receive coils, which excite and detect the MR signal. The MR signals are converted into images by a computer attached to the scanner. Imaging of any part of the body can be obtained in any plane [14]. It is common to use MRI when diagnosing or determining the correct treatment for cancer [5]. The following subsections on MRI are based on [15] unless stated otherwise.

2.2.1 Fundamental physics for MRI

Hydrogen protons are positively charged and have spin that produces a small magnetic field. Hydrogen protons are present in water, which make up about 70% of the human body. The hydrogen protons in the body all have random orientations, thus cancel each other out under normal circumstances. However, if a human body is placed inside a strong magnetic field, such as the ones produced by an MRI scanner, the hydrogen protons will align in the direction of the field. Some will align in the opposite direction and a small majority will align in the same direction as the magnetic field. This creates a small net magnetisation in the direction of the magnetic field. It is this magnetisation that MRI scanners utilize to produce images of the body.

The protons do not align perfectly parallel to the magnetic field, but instead precess around an axis with the same direction as the field. The precession frequency of a proton is defined by a constant, called the gyromagnetic ratio, times the strength of the magnetic field:

$$f = k \times B_0$$

The gyromagnetic ratio for hydrogen protons is 42.6 MHz/T (megahertz per tesla). The magnetic fields used in MRIs are between 1.5T and 3.0T, for comparison the earth's magnetic field is about 0.30×10^{-4}T at the equator and 0.60×10^{-4}T at the poles [16].

Protons in a magnetic field have the ability to absorb and re-emit energy of the same frequency as the proton's precession frequency. The net magnetisation of protons in a magnetic field points in a direction parallel to the main magnetic field. However, by transmitting pulses of energy at the same RF as the precession frequency the energy is absorbed and the net magnetisation rotates away from the field's direction. The rotation angle depends on the strength and duration of the RF pulse.

2.2.2 T1 and T2 weighed images

The direction parallel to the magnetic field is called the longitudinal direction and corresponds to the head-to-toe direction for the patient when they are placed into an MRI scanner. If the longitudinal direction corresponds to the z direction in a coordinate system, one can imagine that the patient's left-right direction corresponds to the x direction, and the front-back direction corresponds to the y direction. This x-y plane is usually called the transverse plane.

By rotating the net magnetisation by 90°, the magnetisation in the longitudinal direction becomes zero and the magnetisation is moved into the transverse plane. After removing the RF signal the magnetisation in the longitudinal direction will begin to grow back. This is called longitudinal relaxation, also known as T1 relaxation.

The hydrogen protons in different tissues have different T1 relaxation rates. This is the source of contrast in T1-weighed images. There will be contrast between different tissue in an image taken when the relaxation of different tissue are at different stages. Tissue with short relaxation time will be brighter than tissue with long relaxation time. T1-weighed images are good at showing the boundaries between different tissue [14].

T2 relaxation also begins by rotating the net magnetisation into the transverse plane. During the RF transmission the protons precess together, meaning they are in phase. After the 90° RF pulse is ended the protons will begin to dephase. Hydrogen protons in different tissue dephase at different rates. This is the source of contrast in T2 weighed images. Tissue with short T2 relaxation time will be darker than tissue with long relaxation time. T2-weighed images are good at showing collections of abnormal fluid [14].

2.2.3 Diffused weighed images

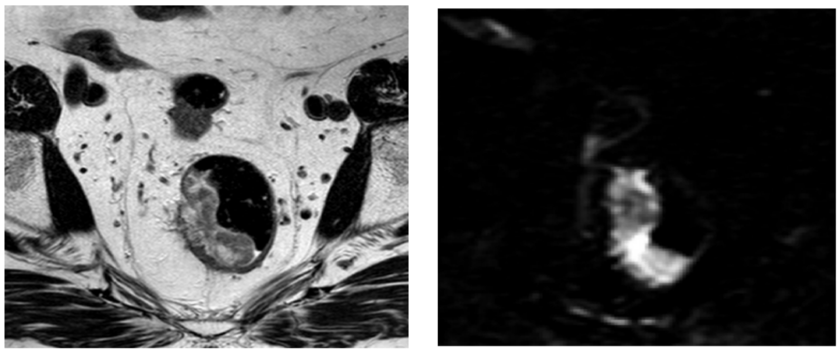
Diffused weighed imaging (DWI) is an MRI technique where the source of contrast comes from differences in the mobility of protons between tissues. Tissues that are highly cellular, such as tumour tissues, restrict the apparent diffusion of water protons.

The sensitivity of the imaging can be altered by changing what is known as the b value:

$$b = k^2 \times G^2 \times \delta^2(\Delta - \frac{\delta}{3})$$

where k is the gyromagnetic ratio, G is the diffusion gradient amplitude, δ is the gradient diffusion length and Δ is the diffusion time. DWIs with different b values can be used to detect and characterise tumours based on the differences in water diffusivity between images, or the images can be qualitatively assessed individually [17].

Figure 2.1 shows a T2-weighed image, Figure 2.1a, and a DWI, Figure 2.1b, of the tumour of one of the patients from the data set used in this thesis.



(a) T2-weighed image of tumour. (b) DWI of tumour.

Figure 2.1: T2-weighed image (a) and DWI (b) showing the tumor of a patient included in the Hypoxia-mediated Rectal Cancer Aggressiveness (OxyTarget) study.

2.3 Machine learning

Machine learning is a branch of artificial intelligence that focuses on self-learning algorithms that find information from and make connections within data. There are three main categories within machine learning: supervised, unsupervised, and reinforcement learning.

In supervised learning, algorithms have access to outcome labels for the data. By comparing these outcome labels with their own output they can evaluate their own performance and provide direct feedback. Supervised learning algorithms use this feedback to maximise a reward function that works to move the output labels closer to the real outcome labels. Supervised learning is used for classification and regression problems [18].

Unsupervised learning algorithms do not have access to outcome labels and must therefore work without direct feedback. Unsupervised learning explores the structure of data without known target variables. It is used to find hidden structures in data, such as clusters or groups.

Reinforcement learning uses a reward system to guide the algorithm through a decision process in order to learn a series of actions. An example of this is teaching a chess program the correct series of actions to win a game [18].

The following subsections on machine learning are based on [18] unless stated otherwise.

2.3.1 Classification

In this thesis classification algorithms will be applied to patient data in the hope of building a model that can correctly predict treatment outcome for patients with colorectal cancer. Classification is an application of supervised machine learning where the goal is to predict the class labels of new instances based on past observations.

2.3.2 Preprocessing

2.3.2.1 Standardisation

Standardisation is a type of feature scaling which gives the data the properties of a standard normal distribution. This means that each feature has a mean centered at zero and a standard deviation of one. This is achieved by subtracting the sample mean from every sample and dividing by the standard deviation as shown in the following equation:

$$x'_j = \frac{x_j - \mu_j}{\sigma_j}$$

where x_j is the observation vector, μ_j is the mean and σ_j is the standard deviation. Data should generally be standardised if it contains features in different ranges. This is so that the models do not become biased towards features with big ranges.

2.3.2.2 Challenges with high-dimensional data

It is easier for an algorithm to find a separating hyperplane between training samples in a sparse high dimensional feature space compared to in a dense or low dimensional feature space. In this way using a simple classifier in a high dimensional space corresponds to using a complex model in a lower dimensional space.

Feature selection and feature extraction are two methods for reducing the feature space. Feature selection methods select a subset of the total features, while feature extraction methods derive information from the original features to construct new features.

Reducing the feature space can improve computational efficiency and reduce generalisation error by removing irrelevant features and noise. It is especially helpful for algorithms that don't support regularisation such as K-nearest neighbors.

2.3.2.3 Repeated Elastic Net Technique

Repeated Elastic Net Technique (RENT) is a feature selection technique that focuses on feature selection stability [19]. It is based on an ensemble of elastic net regularised models of the same type trained on different subsets of the complete dataset. The importance of a feature can be acquired from the frequency at which it is selected across the models. However, even though a feature is selected by one or many models it does not mean that the feature is stable. The feature weights might only be slightly different from zero or have alternating signs across models. To ensure that the selected features are stable, RENT lets the user define the cutoff value for three criteria to optimally exploit the feature weight distribution,

1. τ_1, the minimum required percentage of non-zero feature weights across the models.
2. τ_2, the minimum required proportion of the weights across models to have the same sign.
3. τ_3, the confidence level used in the Student's t-test with rejection of the null hypothesis, assuming that the mean of the feature weights across all models is zero.

The weight distribution across models for a feature must meet all three criteria in order for the feature to be chosen by RENT.

The algorithm used by RENT for binary classification problems is logistic regression models penalised by elastic net. Elastic net consists of a L1 regularisation term $\lambda_1(w) = |w|$, a L2 regularisation term $\lambda_2(w) = ||w||_2$, a mixing parameter $\alpha \in [0, 1]$, where $\alpha = 1$ would mean pure L1 regularisation and $\alpha = 0$ would mean pure L2 regularisation, and the parameter γ which defines the regularisation strength. Altogether, the elastic net regularisation term is defined as

$$\lambda_{enet}(w) = \gamma[\alpha\lambda_1(w) + (1 - \alpha)\lambda_2(w)]$$

Logistic regression is explained in section 2.3.4.2, while L1 and L2 regularisation are explained in section 2.3.3.5.

The purpose behind choosing stable features is to strengthen model interpretation and increase model robustness [19]. Figure 2.2 shows the τ_1 values of the features from a RENT analysis on a subset of the Wisconsin breast cancer dataset from the UCI database. For more information about this dataset see [20].

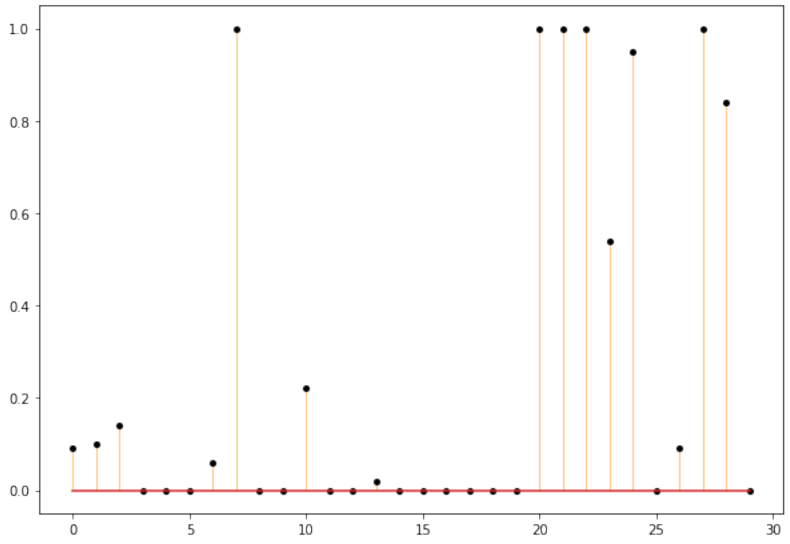


Figure 2.2: Barplot of τ_1 values from a RENT analysis performed on a subset of the Wisconsin breast cancer dataset from the UCI database. The feature indexes are on the x-axis, and the y-axis shows the τ_1 values.

The figure shows that some features, such as the features with index 7 and 20, had nonzero feature weights across all models in the ensemble, while others, such as the features with index 4 and 5, were never assigned a weight unequal to zero. There were also some features that were only selected by elastic net for a few of the models, such as the features with index 3 and 10. These features would be considered unstable, since whether they were selected or not was dependent on the subset used used for training.

2.3.2.4 Principal component analysis

Principal component analysis (PCA) is an unsupervised feature extraction method. PCA reduces a d dimensional feature space into k dimensions by constructing a $d \times k$-dimensional transformation matrix that make it possible to map vectors from d to k dimensions.

The correlation matrix of a dataset X is X_{corr}. The transformation matrix for X is constructed out of the first k eigenvectors of X_{corr}.

Geometrically, PCA works by first finding the vector in the feature space that contains the most variation within the observations. This becomes the first component. It then finds the second component by looking among the vectors that are orthogonal to the first component and chooses the one with the most variation, and so on. Each principal component is orthogonal to all others, thus there is no correlation between them.

While PCA finds the vector with the most variation within the observations, it does not take into account the observation classes because it is an unsupervised method. This means that observations with different classes can end up being mapped to similar locations in the new vector space.

2.3.3 Training process

2.3.3.1 Train and test set

In machine learning, it is common to assign the data matrix the name X and the corresponding response vector the name Y. As the goal of a classification model is to predict the class of unseen data, the dataset is divided into a training dataset and a test dataset. The training dataset is used to train the model, while the test dataset is used after training to evaluate how well the model generalises to unseen data.

During model selection, e.g. hyperparameter tuning, each model needs to be evaluated on unseen data. If the same test set was used to evaluate every model it would cease to be unseen data and become part of the training data. In other words, instead of finding a model that generalises well to any potential test set, the process is finding a model that fits a specific test set and are likely to overfit. To avoid this problem we need to change the evaluation method of the models.

2.3.3.2 Cross validation

Cross validation (CV) is a group of model validation methods where the averages over multiple fitness estimates are used to evaluate a model. In k-fold CV the dataset is randomly divided into k folds where $k - 1$ folds are used to train the model and the remaining fold is used for performance evaluation. The training and evaluation process is repeated k times and each fold is used as the evaluation set once, see Figure 2.3. Thus k models and performance evaluations are obtained. By taking the average across the folds, a more accurate and robust performance evaluation is achieved compared to using a single training and test set, which would only give a single evaluation.

Because multiple folds are used as training and validation at different points during the process, the results are less likely to be affected by unlucky splits. However, unlucky splits can still occur if there is little data available and few splits. An unlucky split would be a split of the data where the subsets are not representative of the whole dataset. For example, a model trained on a training set only made up of easily classified samples and evaluated on a test set that contains difficult to classify samples would have a much lower performance evaluation compared to a model that was trained and evaluated on sets that were the other way around.

It is also useful to ensure that each class is represented in each fold. Both for reasons of class imbalance, as stated later in section 2.3.3.4, and also because a model can not be trained or tested on a class if it is not represented at all in a fold. Stratified k-fold CV is a variation of k-fold CV which strives to preserves the class ratio of the dataset when making the folds.

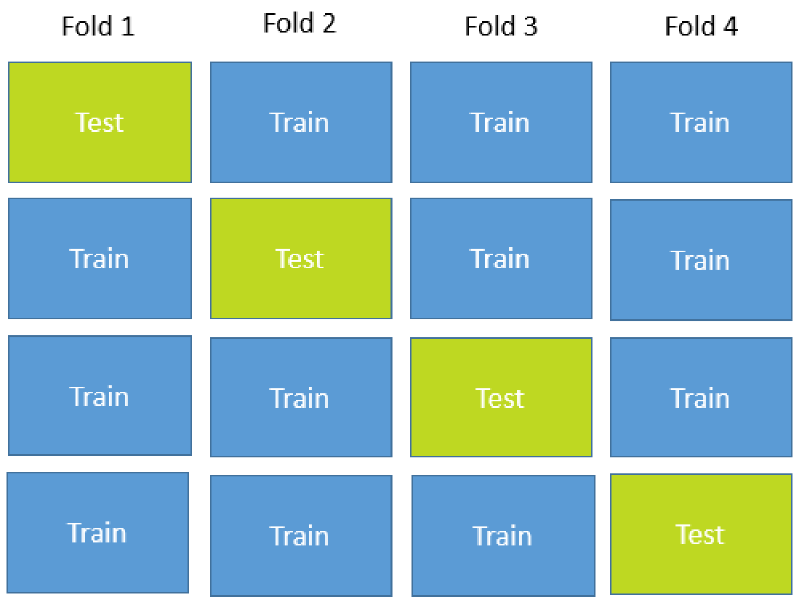


Figure 2.3: An example of 4-fold CV.

2.3.3.3 Over- and Underfitting

One of the reasons for testing the model on unseen data is to check for over- or underfitting. Over- and underfitting are related to bias and variance. Variance is the variability in the predictions that a model gives for a data point. Bias is the systemic difference between the correct value and the predictions given by a model for a data point. A model with low bias and low variance will give accurate and concentrated predictions, while a model with high bias and high variance will give inaccurate and spread out predictions. In machine learning the optimal model is one with low bias and low variance, but this is difficult to achieve in reality. This is because bias and variance are a trade-off.

A model with low bias is often the result of overfitting. The model has too many parameters, is overly complex and specialised on the particulars of the training data. A model that has low variance is often the result of underfitting. The model has too few parameters resulting in a model that is overly simple and unable to capture the underlying pattern in the data. Both of these types of models are unable to generalise well to unseen data. To find a good model one would need to find the balance between bias and variance that minimises the total error.

One way to detect over- or underfitting is to compare the prediction results from the training set and test set. A model that performs much better at predicting class labels on the training samples compared to the test samples is showing signs of overfitting, while an underfit model will predict training samples and test samples about the same, but both will give poor predictions.

2.3.3.4 Class imbalance

When the number of samples from each class in a dataset differ substantially, the dataset is considered imbalanced. An imbalanced dataset can influence the training and validation process. The dataset used in this thesis has a class imbalance where approximately 30% of samples belong to class 1, while the remaining 70% belong to class 0. A classifier could thereby simply classify all samples as class 0 and still obtain an accuracy of 70%, even if this would mean misclassifying all samples from class 1.

A situation like this could cause the classifier to become biased towards the majority class. There are several methods to mitigate this. One could use a random subset of the majority class of equal size to the minority class, sample the minority class with replacement until the classes become of equal size, generate synthetic samples for the minority class, or introduce regularisation or class weighing.

2.3.3.5 Regularisation

Regularisation is a way to put constraints on a model, usually in order to regulate complexity. Two popular regularisation methods are L1 and L2 regularisation. The L1 norm of the feature weight vector w is given by

$$L1 : \| w \|_1 = \sum_{j=1}^{m} | w_j |$$

and the L2 norm of the feature weight vector is given by

$$L2 :\| w \|_2^2 = \sum_{j=1}^{m} w_j^2$$

where w_j is the weight corresponding to the jth feature in the data and m is the total number of features. The purpose of L1 and L2 regularisation is to punish large feature weight values, as large feature weights is a sign of overfitting. L1 and L2 regularisation is implemented by adding the terms above to the loss function of a machine learning algorithm, together with a parameter that controls the regularisation strength. Depending on the regularisation strength. L1 regularisation can yield sparse feature vectors by pushing most weights to zero. In this sense, it can be considered a feature selection technique.

2.3.3.6 Hyperparameter Tuning

Most machine learning algorithms have two types of parameters: those that are learned through training, such as the feature weights in logistic regression, and those that are defined together with the model, such as regularisation strength in logistic regression. This last type is also called hyperparameters. When selecting an algorithm it is often advantageous to try out different hyperparameter combinations. This makes hyperparameter tuning a part of model selection.

Parameter tuning means to find the best hyperparameter combination. To tune a model, one defines multiples of the same type of model with different values for the hyperparameters one wishes to tune, e.g. logistic regression models with different regularisation. Tuning can be done by performing a grid search. Grid search is a brute-force method that compares all the possible combinations of the defined hyperparameters to find the optimal.

2.3.4 Classification algorithms

2.3.4.1 Support Vector Machine

Support vector machines (SVMs) are a set of supervised learning methods with applications within classification, regression and outliers detection [21]. The goal of SVMs is to maximize the distance between the decision boundary and the closest training samples. This distance is called the margin, and the closest training samples are called support vectors, see Figure 2.4. The benefit of having large margins is that it lowers the generalization error, as models with small margins are prone to overfitting.

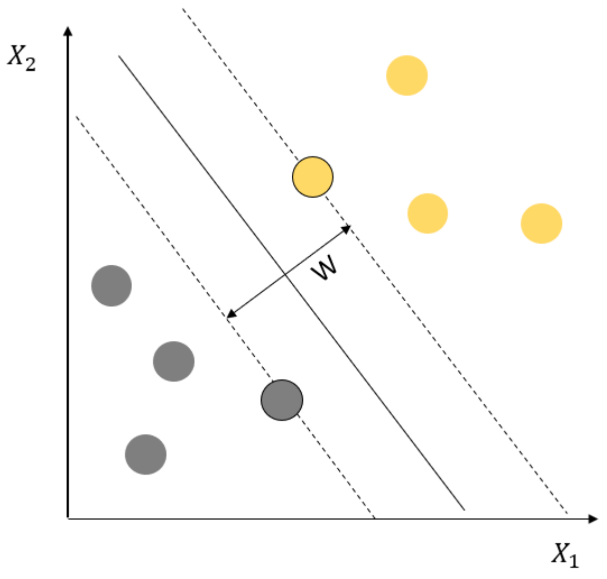


Figure 2.4: A schematic showing the margin. The circles represents samples from two different classes, the circles on the dotted lines are support vectors and w is the margin.

Support vector machines are effective in high dimensional spaces, even in cases where the number of dimensions is higher than the number of samples. However, if the number of dimensions outnumber the samples it is important to avoid overfitting by implementing different kernel functions and regularisation [21].

2.3.4.2 Logistic Regression

Despite its name, logistic regression is not a regression algorithm, it is instead a classification algorithm. Logistic regression models are used for binary classifications problems such as pass/fail or healthy/sick.

Logistic regression is based around the logistic function, also known as the sigmoid function. The sigmoid function is a S-shaped curve that can map any real number into a value between 0 and 1, but never exactly 0 or 1. It is written as:

$$\theta(z) = \frac{1}{1 + e^{-z}}$$

The input of the sigmoid function, z, is the linear combination of weights w and sample features x, $z = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m$, where w_0 is the bias unit and x_0 is equal to 1. The output from the sigmoid function is interpreted as the probability that a sample belongs to class 1 given the sample's features x. The class probabilities are converted into a binary class prediction via a threshold function. One of the reasons why logistic regression is so widely used is because it gives class probabilities in addition to class predictions.

The cost function used for training the weights in logistic regression is the log-likelihood function:

$$J(w) = \sum_{i=1}^{m} [-y^i log(\theta(z^i)) - (1 - y^i) log(1 - \theta(z^i))]$$

where y^i is the true class label for sample i. Logistic regression models are less complex compared to SVMs, making them easier to train and update. However, logistic regression models are more vulnerable to outliers since they try to maximise the conditional likelihoods of the training data, while SVMs care more for the support vectors by the decision boundary.

2.3.4.3 Decision Trees

Decision tree classifiers are a group of models that break down data by asking a series of questions. During the training process a decision tree learns which questions to ask in order to best classify the data. Decision trees are effective on both categorical and numeric variables. Decision trees are especially useful if interpretability is important because the rationale behind each classification is apparent in the structure of the tree.

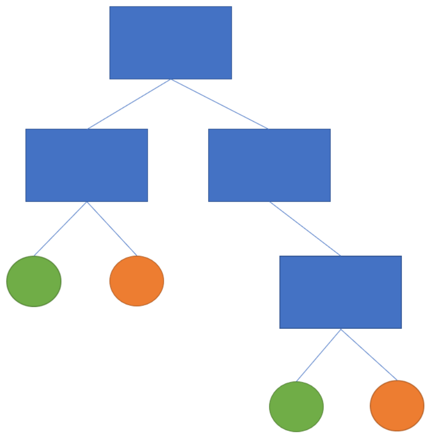


Figure 2.5: A schematic of a decision tree. The rectangles represent nodes and the circles represents leaves.

As seen in Figure 2.5, a decision tree has one root and a number of nodes. Each node is connected to its parent node. The nodes without children are called leaves.

At each node one can calculate the information gain. The information gain is the difference between the impurity of the parent and the sum of the impurity of the child nodes. For binary trees, each parent node is split into two child nodes. The information gain is expressed with the following equation:

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

where I is the measure for impurity, D_p is the dataset of the parent, D_{left} and D_{right} are the datasets of the left and right child node, respectively, N_p is the number of samples at the parent node, and N_{left} and N_{right} are the number of samples at the left and right child node. f is the feature to perform the split.

The steps used to build a decision tree goes like this, start at the root and split the data on the feature that leads to the biggest information gain. This creates two child nodes. Move down to the child nodes and split again on the feature that leads to the biggest information gain. Repeat this process until all nodes are pure, i.e. only contain data from one class. This process can lead to very deep trees that can easily overfit, so in practice it is normal to prune the tree by specifying the maximal depth. Feature scaling is not required for decision trees.

2.3.4.4 Random forest

A random forest is an ensemble of decision trees. The idea behind random forest is that training a number of deep decision trees that individually suffer from overfitting and taking the average will lead to a more robust model. The random forest algorithm can be described in the following steps:

1. Draw a random bootstrap sample of size n
2. Grow a decision tree from the sample. At each node:
    (a) Randomly select d features without replacement.
    (b) Split the node using the features that provides the best split
3. Repeat steps 1 and 2 k times.
4. Aggregate the predictions by each tree to assign the class label by majority vote.

The splitting process is slightly different in a random forest compared to a single decision tree. Instead of seeking out the best split among all the features, only a random subset is considered.

The advantage of random forest over a single decision tree is robustness. For hyperparameter tuning, overfitting of individual trees are no longer of much concern. Therefore the trees no longer need to be pruned by defining a maximum depth, but a new parameter k, which controls the number of decision trees in the ensemble, has been introduced. Typically, the greater the number of trees, the better the estimated performance of the model will be, at the cost of increased computational expense.

2.3.4.5 K-nearest neighbours

K-nearest neighbor (KNN) is a non-parametric machine learning algorithm. This means that instead of estimating parameters from the training data to classify new data, it memorises the training data. The KNN algorithm can be summarised in the following steps.

1. Choose the number of nearest neighbors to base prediction on, k, and a distance metric.
2. Find the k nearest neighbors to the sample that is being classified.
3. Assign the class label by majority vote.

In other words, KNN finds the k nearest, meaning most similar, samples to the sample that is being classified and determines the class by majority voting. The main advantage of non-parametric algorithms is that they can immediately adapt to new data. The downside is that computational complexity grows linearly with the number of samples in the training data and none of the training data can be discarded.

2.3.5 Ensemble learning

Ensemble learning methods combine multiple classifiers into one single meta-classifier. The majority vote classifier is a popular ensemble learning method where the class label prediction of the meta-classifier is decided via combining the predicted class label from each classifier and selecting the class label with the most votes.

Ensemble methods have two main advantages: performance and robustness. By combining models, better performance is archived because the amount of error in the predictions that is due to variance is reduced, and the spread of the prediction scores for the ensemble will be smaller than the spread for each individual model in the ensemble, leading to a more robust classifier.

However, an ensemble will not always lead to better performance. If the ensemble is made up of one well-performing model and a group of under-performing models, the ensemble may only perform as well as the one top-performing model. I could also perform worse than the top-performing model if the performance of the ensemble is lowered by the under-performing models. In such instances, it is better to simply use the top-performing model instead of the ensemble.

2.3.6 Evaluation

There are many different methods and metrics for measuring the performance of a classification model. The metrics chosen for this thesis are presented in this section.

2.3.6.1 Confusion matrix

A confusion matrix is a square matrix that reports the counts of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) in the predictions of a classifier. It is one way to visualise and analyse the performance of learning algorithms.

2.3.6.2 Simple evaluation statistics

From the confusion matrix one can calculate a number of statistics. Among them are accuracy and recall. Put simply, accuracy is the percentage of

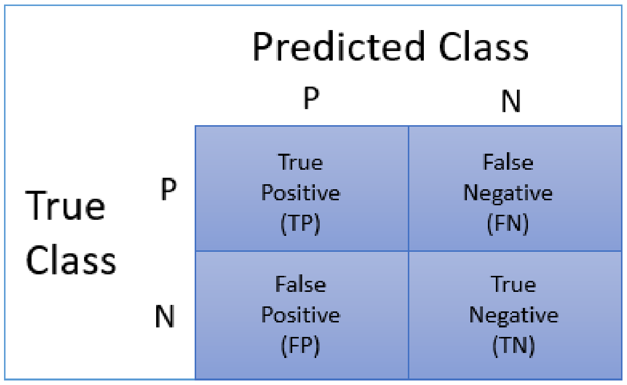


Figure 2.6: Confusion matrix showing TP, FP, TN, and FN.

samples that were labeled correctly as seen below:

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

Recall is the percentage of positive samples that were labeled correctly as seen below:

$$REC = \frac{TP}{TP + FP}$$

2.3.6.3 Matthews correlation coefficient

The Matthews correlation coefficient (MCC), also knows as phi coefficient, is a measure of the quality of binary and multiclass classifications. MCC is given by:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

The MCC is in essence a correlation coefficient value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction [22]. As can be seen from the equation above, MCC takes into account true positive (TP), true negative (FN), false positive (FP), and false negative (FN) counts. This makes it a more balanced measure compared to accuracy and recall when the classes are of different sizes.

Chapter 3

Materials and methods

The data used in this thesis was supplied by the Functional MRI of Hypoxia-mediated Rectal Cancer Aggressiveness (OxyTarget) study [23]. The study's aim was to ”identify novel imaging biomarkers for hypoxia-induced rectal cancer aggressiveness, with the goal of reliably predicting patients with poor response to CRT and high risk of poor metastasis-free survival at time of diagnosis” [24]. The study started in 2013 and enrolled 192 patients with colorectal cancer at Akershus University Hospital between October 2013 and December 2017. The study recruited all patients with colorectal cancer during this period, thus the data contains both patients that received CRT treatment and patients that did not. The study concluded in 2020.

MRIs of the patients were taken before any treatment had started by a Philips Achieva 1.5T machine from Philips Healthcare, Best, The Netherlands. DWIs were acquired for seven different b-values: 0, 25, 50, 100, 500, 1000 and 1300 s/mm^2. T2-weight images had voxel size $0.35 \times 0.35 \times 2.50\, mm^3$, and the DWIs had voxel size $1.25 \times 1.25 \times 4.00\, mm^3$ [25].

Of the 192 patients enrolled in the study, as many as 111 were excluded from this thesis due to various reasons such as rectal cancer not being histologically confirmed, not meeting standards for image acquisition or quality, or other problems during image acquisition and processing [26] [25]. This leaves a total of 81 patients to be analysed.

The data consists of three files. The first file contains clinical data, while the other two contain information extracted from MRIs. The difference between the MRI datasets is the voxel resolution of the images that the information was extracted from. The set that from here on will be referred to as set 1 used images with the original resolution, while set 2 used images that had been resampled to voxel resolution $1 \times 1 \times 1\, mm^3$ [25].

Information about the set 1 and set 2 data files and the information extraction process known as radiomics is described in sections 3.1 and 3.2. Information about, and the preprocessing of, the clinical data is described in section 3.3.

3.1 Radiomics

The process of converting medical images into high dimensional quantitative data is known as radiomics [4]. Radiomics is based on the assumption that medical images contain information that is not apparent through visual inspection alone [27].

Before feature extraction can begin, a region of interest (ROI) for two-dimensional approaches or a volume of interest (VOI) for three-dimensional approaches must be defined. These regions define the area of the image where features are extracted from. Another important step is to group the original intensity values into specific range intervals. This process is called binning. Binning normalises the images and is thought to make radiomic features more reproducible across different samples, especially when used on data with arbitrary intensity units such as MRI [27].

A number of different features can be extracted from medical images, below is a short explanation of the types of features seen in set 1 and set 2.

3.1.1 First-order statistics features

First-order statistics features describe the distribution of individual voxel intensities within the ROI. First-order statistics features describe the distribution without concern for spatial relationships, such as mean, median, maximum and minimum values [**radi˙fact˙and˙challenges**].

3.1.2 Shape features

Shape features describe the shape of the ROI and its geometric properties. They are based on the binary mask used for segmentation of the ROI and are thus independent of gray values [28]. Shape features may be three-dimensional or two-dimensional. Examples of shape features are volume and sphericity [**radi˙fact˙and˙challenges**].

3.1.3 Texture features

Texture features describe the relationships between voxels with similar or dissimilar contrast values in the ROI and can measure aspects of a tumour like heterogeneity, asymmetry and contrast [4].

3.2 MRI data files

The radiomics features were extracted by Aase Langan as part of her thesis *MRI-Based Radiomics Analysis for Predicting Treatment Outcome in Rectal Cancer* [25]. All the images were binned with the intensity range interval set to 25 and used the same tumor delineation, ROI. The features were extracted using the Biorad software. Biorad is a tool for extracting radiomic features that also allows the user to perform machine learning experiments [29]. Biorad is based on the python package pyradiomics, whose documentation can be found here [28].

The two MRI datasets, named set 1 and set 2, contain first-order statistics, shape features and texture features from both T2-weighted images and all the DWIs (b0 - b6). Shape features from DWIs were only included once. Each set contains in total 81 patients and 772 features. A detailed explanation of the extracted features can be found in the pyradiomics documentation and Aase Langan's thesis.

3.3 Data selection

As the goals of this thesis are related to predicting treatment outcome for colorectal cancer with overall survival as the prediction endpoint, four patients with listed cause of death unrelated to colorectal cancer were removed from all datasets. This reduced the total number of patients from 81 down to 77.

No further preprocessing of set 1 or set 2 was deemed necessary. However, the clinical dataset included features judged to be superfluous and missing values.

In the first step of preprocessing the clinical data, data exploration, one patient was removed because they were identified as an outlier. After this step the data had dimensions (76, 97).

The second step was to remove features that were judged to be irrelevant to the modeling process. These features totaled 42 and consisted mostly of dates for medical procedures, hospital data such as patient identification numbers and doctor's comments. At this time the data had dimensions (76, 55).

The third step was to use one-hot encoding to change the binary categorical variables into ones and zeros in order to make the data to be usable in the machine learning algorithms. This did not result in any changes to the dimensions of the data.

The fourth step was to tackle the missing values in the dataset. Only 11 of 55 features did not contain any missing values, and all patients had at least one missing value.

The missing values were dealt with by tallying up the number of missing values in each column and defining a cutoff value. If the total number of missing values in a column exceeded the cutoff value, the column was removed from the dataset. After removing all columns that contained a larger number of missing values than the cutoff, all rows that still contained missing values were removed. The different cutoff points listed in Table 3.1 were experimented with. In the end the cutoff value was set to five because keeping patients was deemed more important than keeping features.

Table 3.1: Values used as cutoff point for the acceptable number missing values in a column and the resulting data dimensions.

Cutoff value	**Resulting dimensions**
0	(76, 11)
5	(58, 20)
10	(41, 33)
15	(32, 37)
20	(27, 44)

The final clinical dataset included 58 patients and 20 features. Table 3.2 lists the remaining features in the dataset after preprocessing and Table 3.3 lists the mean value and distribution for some of the features in the clinical dataset.

Table 3.2: The features of the clinical dataset after preprocessing. The table states the names of the features, and definitions or other additional information about the features are provided in the explanation column.

Feature name	**Explanation**
Gender	Binary: male or female
BMI	Body mass index (kg/m^2)
Age	Age at time of referral
Distance from anal opening	Placement of the tumour, distance in centimeter from the anal opening measured by a rigid scope
MSI	Microsatellite instability
Distance from anal opening on MRI	Placement of tumour, distance in centimeter from anal opening measured on MRI
mrT (TNM ed.7)	T-stage
mrN	N-stage
Suspicion of metastasis at time of diagnosis	Binary
Stage	0 = T0N0, 1 = T2N0, 2 = T3N0, 3 = TXN1-2, 4 = TXNXM1
Preoperative CRT	Whether or not a patient received CRT treatment
CEA baseline	From blood test
CRP baseline	From blood test
Hb baseline	From blood test
Leukocytes	From blood test
Sodium	From blood test
Potassium	From blood test
Creatinine	From blood test
Bilirubin	From blood test
Adjuvant treatment	Binary

Table 3.3: Summary over some of the features in the clinical dataset. The listed values for BMI, age and CEA baseline is the mean across the patients.

Number of Patients	58
Gender	34 male, 24 female
BMI	25
Age	63
Stage	
1	8
2	17
3	19
4	14
Preoperative CRT	27 yes, 31 no
CEA baseline	34
OS-event	18 yes, 40 no

3.4 Splitting the datasets

During this thesis set 1, set 2 and the clinical data were each divided into three subsets. One subset contained the entire patient cohort, in this case 77 patients, one subset only contained patients who received CRT treatments and one subset only contained patients who did not receive CRT treatment. These different subset groups will from here on be referred to as the CRT subsets, the no-CRT subsets and the all-patients subsets.

The clinical all-patients subset contained 58 patients and 20 features. The clinical CRT subset contained 27 patients, while the no-CRT subset contained 31 patients. A summary of the clinical CRT and no-CRT subsets are listed in Tables 3.4 and 3.5.

The set 1 and set 2 all-patients subsets contained 77 patients and 772 features, while the set 1 and set 2 CRT subsets contained 34 patients and the no-CRT subsets contained 43 patients. Note that all the subsets of set 1 and set 2 contained more patients than the corresponding clinical data subset. This is because some patients were removed from the clinical data due to missing values.

Table 3.4: Summary of the clinical CRT subset.

Number of Patients	27
Gender	17 male, 10 female
BMI	26
Age	59
Stage	
1	0
2	6
3	15
4	6
CEA baseline	28
OS-event	9 yes, 18 no

Table 3.5: Summary of the clinical no-CRT subset.

Number of Patients	31
Gender	17 male, 14 female
BMI	25
Age	67
Stage	
1	8
2	11
3	4
4	8
CEA baseline	40
OS-event	9 yes, 22 no

3.5 Workflow

The workflow seen in Figure 3.1 was used to make three majority vote classifiers, one with the CRT subsets, one with the no-CRT subsets and one with the all-patients subsets.

The workflow follows 5 steps. Step 1 is to divide the data into training and test samples. The process behind dividing the data is explained in section 3.6. The training sets are fed into a pipeline that creates the majority vote classifier, and the test samples are used to evaluate the classifier as part of step 4.

In step 2, PCA and RENT are performed on each of the training sets. PCA and RENT transform and reduce the datasets, respectively. One version of each of the training sets goes through step 2 unchanged.

In step 3, four different models are trained on the original training datasets and the datasets created by PCA and RENT. The models are one KNN, one logistic regression, one SVC and one random forest model. This results in 36 models: 12 from set 1, 12 from set 2 and 12 from the clinical data.

All models utilise hyperparameter tuning by performing a grid search over a pre-specified parameter range. Grid search utilises k-fold CV to assess and compare different parameter combinations. This means that each combination is trained and tested k times on different subsets. Grid search calculates the mean train and test score across the folds, as well as mean standard deviation values for each combination. The hyperparameter combination with the highest mean test value is deemed the best combination. The metric used to assess the models during hyperparameter tuning is MCC.

In step 4, the model with the highest mean test score value from each set are combined into a majority vote classifier. This means that the classifier consists of three model that predicts samples based on information from either set 1, set 2 or the clinical data. The majority vote classifier predicts new samples by first having each of the three models make a class prediction based on their respective dataset. Each prediction counts as one vote for the predicted class. The class that has the most votes will be the classifier's class prediction. In step 5, the majority vote classifier is used to predict the class of the test samples.

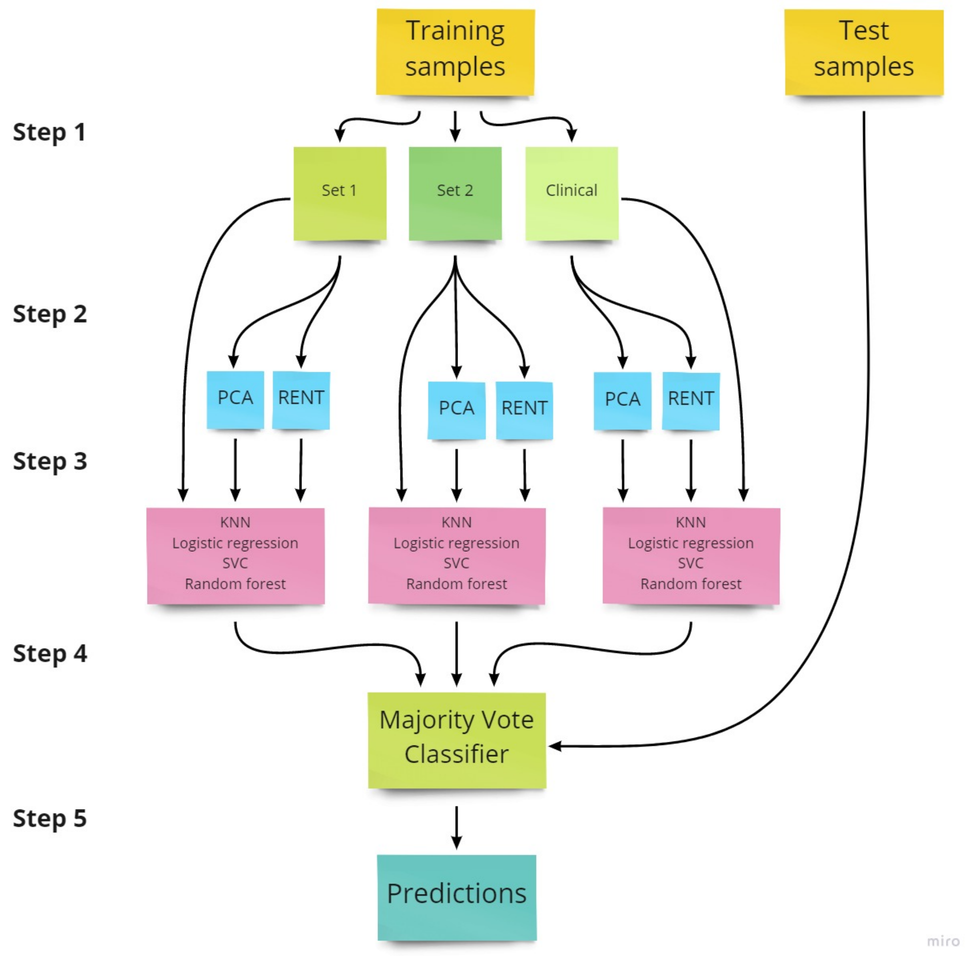


Figure 3.1: The work flow for building the majority vote classifier, and the class prediction process.

3.6 Validation

In this thesis an outer K-fold CV was used for the entire data during step 1 of the workflow. In addition, an inner K-fold CV was used whilst training RENT and when performing grid search on the hyperparameters of the models in step 2 and 3.

The outer K-fold CV used four folds. Recall that all the clinical subsets were smaller than the set 1 and set 2 subsets because some patients had to be removed during preprocessing. Since the test folds were going to be used to evaluate the majority vote classifier at the end of the workflow, the samples in each test fold must be present in all three datasets. This means that the patients that were present in set 1 and set 2, but not in the clinical data could not be used as test data. Therefore, when patients were sampled for the test and training folds, only patients present in the clinical data were part of the selection. In practice this means that some of the patients in set 1 and set 2 were only used as training samples and never as test samples.

Figures 3.2 and 3.3 depict how the datasets were divided into train and test sets for the outer 4-fold CV. All of the patients in the clinical data were used as training samples three times and as test samples one time.

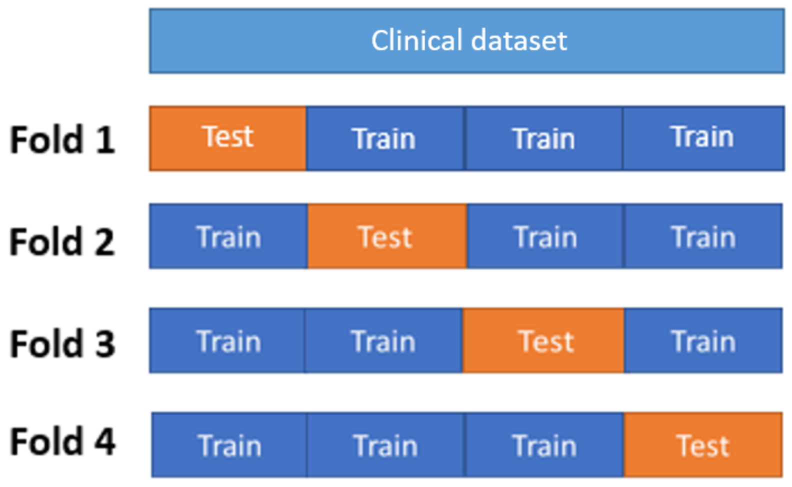


Figure 3.2: Overview of how the clinical datasets were divided into folds.

As for set 1 and set 2, the patients that were also in the clinical dataset were used as training samples three times and as test samples one time. The patients that were not present in the clinical data, called leftover training data in Figure 3.3, were used as training data in all four folds.

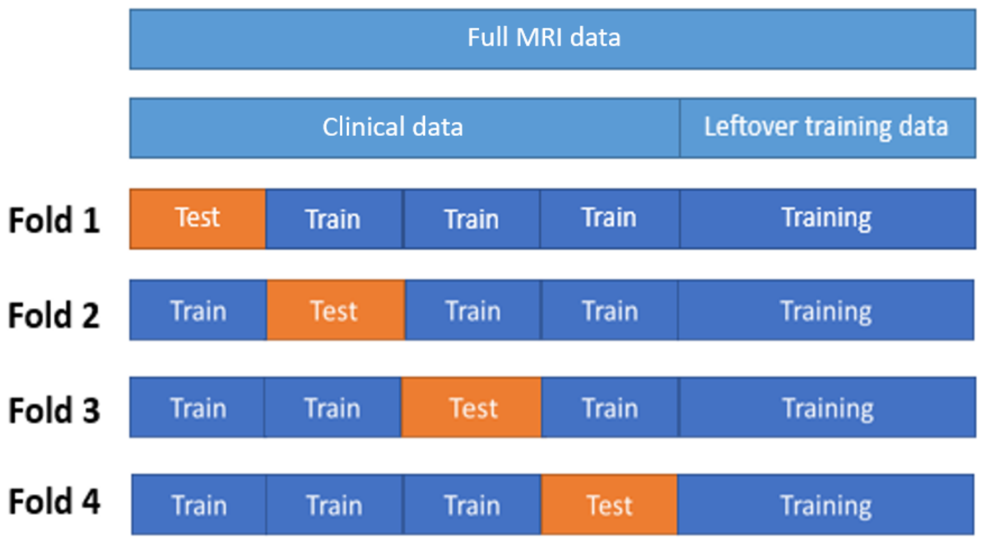


Figure 3.3: Overview of how the set 1 and set 2 datasets are divided into folds. Leftover training data referrers to the samples only present in set 1 and set 2.

3.7 Software

The code used in this thesis was written in Python version 3.8.5 on an Anaconda platform. Modules used were Pandas version 1.1.3, Sklearn version 0.23.2, Numpy version 1.19.2 and Re version 2.2.1.

The python programs used to preprocess the data and build the workflow are included in appendix B.

Chapter 4

Results

When the workflow is being used on a subset group, all-patients, CRT or no-CRT, all the subsets, set 1, set 2 and clinical data, are split into four outer folds that make four training-test set combinations. This means that there are 12 such combinations per subset group.

During each of the outer folds, three training sets, one set 1, one set 2 and one clinical data set from the same subset group, are fed into the pipeline that makes the majority vote classifier while the three corresponding test sets are used to evaluate the classifier.

The first step in the pipeline is to perform RENT and PCA on each of the three training sets. The second step is to train four different models on the untreated sets and the sets that have been treated by RENT and PCA. This means that each training set that was fed into the pipeline results in 12 models. Since three sets were put into the pipeline this means a total of 36 models. Three of these models are selected for the majority classifier, one from each set.

The outer folds are shuffled around so that each training-test set combination is used to build and evaluate the majority vote classifier one time. This means that the process mentioned above that resulted in 36 models is repeated 4 times. Which means that a total of 144 models are trained and considered for the majority vote classifier, for each of the three subset groups. The fact that there are three subset groups means that all in all, a total of 432 models have been created during this thesis.

All the figures in the following sections are based on tables in appendix A. All datasets that were treated by PCA were reduced to 10 principal components. The cutoff values used in RENT are listed in Table 4.1

Table 4.1: The cutoff values used in RENT for all datasets.

Parameter	**Cutoff value**
τ_1	0.20
τ_2	0.20
τ_3	0.80

4.1 Results from all-patients subsets

This section presents outer fold prediction scores from the majority vote classifier, as well as the inner fold prediction scores of the models that made up the classifier, from when the all-patients subsets were fed into the work flow described in section 3.5.

Figure 4.1 shows the outer fold prediction scores obtained in each of the four outer folds described in Section 3.6. In other words, it shows results from step 5 in the workflow. The models based on information from set 1 had generally low MCC-scores, except during fold 3. Two out of four scores were zero, the same as random guessing. The model based on information from set 2 also had generally low scores, except during fold 1 and was the only model with a negative prediction score. A negative MCC score is worse than random guessing. The model based on information from the clinical dataset had generally high prediction scores over all four folds. Lastly, the majority vote classifier had mid-level prediction scores compared to the other three models.

The mean prediction score and standard deviation for the majority vote classifier and the models that made up the classifier are listed in Table 4.2.

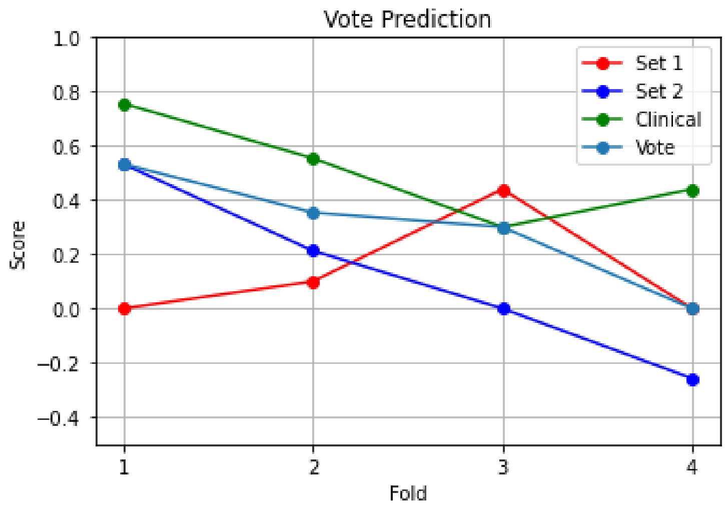


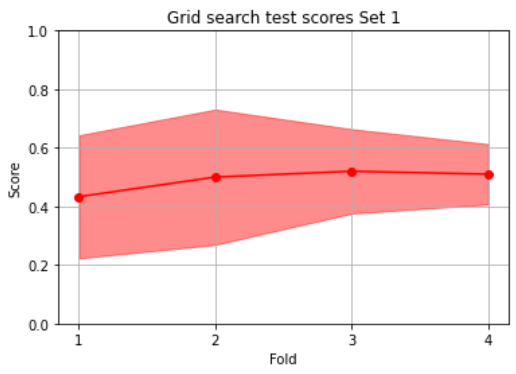
Figure 4.1: Prediction scores from the set 1, set 2 and clinical models that were chosen for the majority vote classifier and prediction scores from the majority vote classifier itself. The graphs display the MCC scores of the models across the four outer folds of the all-patients subsets.

Table 4.2: The mean values and standard deviations from the outer folds of the all-patients subset.

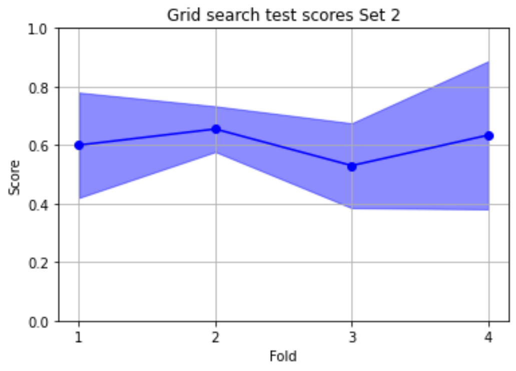
	Set 1	**Set 2**	**Clinical**	**Vote**
Mean	0.135	0.122	0.513	0.30
Std	0.181	0.29	0.167	0.192

Figure 4.2 shows the mean test scores and the mean standard deviation obtained from the inner cross validation performed at step 3 in the workflow for the models selected to be in the majority vote classifier. The mean test scores for all the set 1 models, Figure 4.2a, show little variation between folds, but do show somewhat high standard deviations. All the mean test prediction scores are higher than the scores obtained during the respective outer fold shown in Figure 4.1.

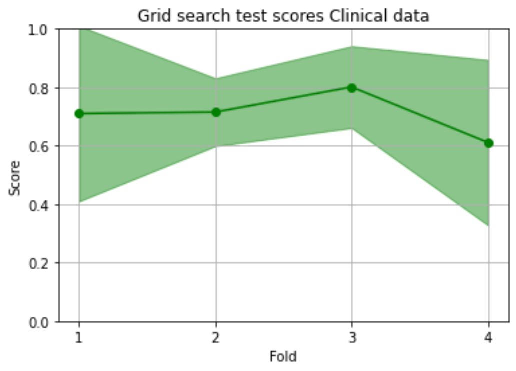
The mean test scores for the set 2 models, Figure 4.2b, also show little variation between folds, but varying degrees of standard deviation. Again, all the mean test prediction scores from the inner folds were higher than the scores obtained during the respective outer folds shown in Figure 4.1



(a)



(b)



(c)

Figure 4.2: Mean test scores with mean standard deviations calculated during grid search for the models selected to be in the vote classifier from each all-patients subset: (a) set 1, (b) set 2, (c) clinical data.

The mean test scores for the clinical data models, Figure 4.2c, again show little variation between folds and varying degrees of standard deviation. The score obtained during fold 1 is similar to that shown in Figure 4.1, while the scores for the other folds are lower.

4.2 Results from CRT subsets

This section presents the outer prediction scores of the majority vote classifier, as well as the inner prediction scores of the models that made up the classifier when the CRT subsets were fed to the work flow described in Section 3.5.

Figure 4.3 shows the prediction scores from the majority vote classifier and the prediction scores of the models that made up the classifier on the test samples from each of the four outer folds described in Section 3.6. All the models follow a similar pattern: fold 1 and 3 have high scores, while fold 2 and 4 have low scores. Only the models based on the clinical dataset stood out. This was the only set that did not have a score higher than zero during fold 1 and also the only set with a score lower than zero during fold 4.

The mean prediction score and standard deviation for the majority vote classifier and the models that made up the classifier are listed in Table 4.3.

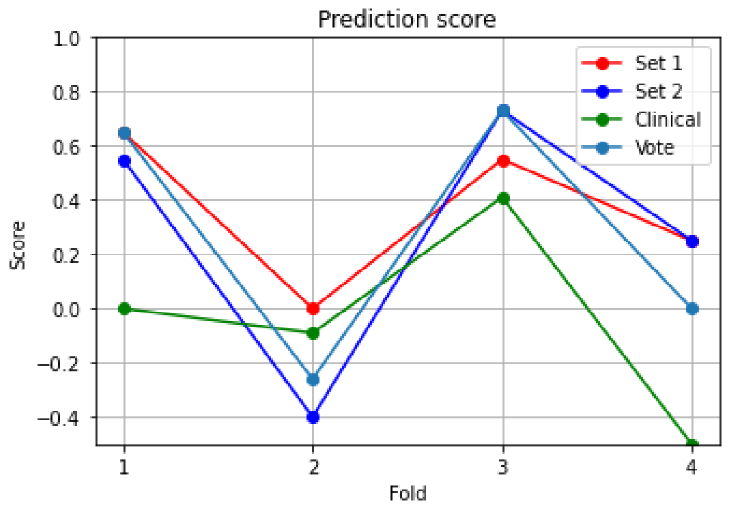


Figure 4.3: Prediction scores from the set 1, set 2 and clinical models that were chosen for the majority vote classifier and prediction scores from the majority vote classifier itself. The graphs display the MCC scores of the models across the four outer folds of the CRT subsets.

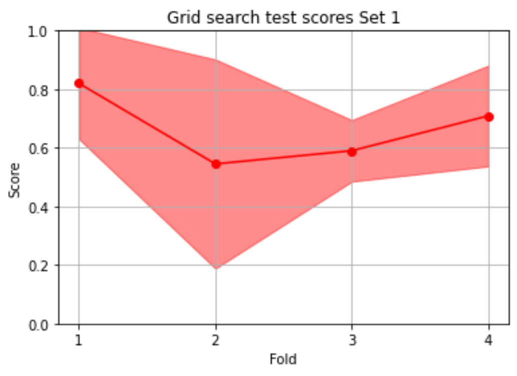
Table 4.3: The mean values and standard deviations from the outer folds of the CRT subset.

	Set 1	**Set 2**	**Clinical**	**Vote**
Mean	0.361	0.282	-0.04	0.279
Std	0.254	0.429	0.326	0.419

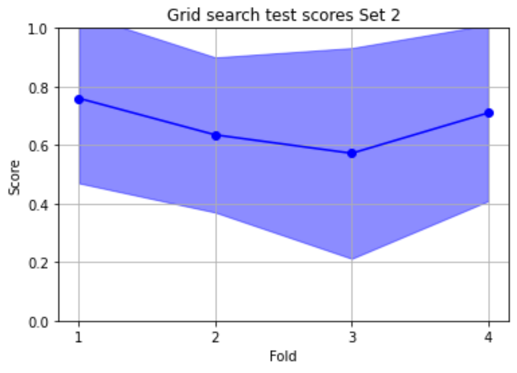
Figure 4.4 shows the mean test scores and the mean standard deviations obtained from the inner cross validation performed during grid search for the models selected to be in the majority vote classifier. The mean test scores for all the set 1 models, Figure 4.2a, show some variation between folds and a very big standard deviation for fold 2. The prediction scores for fold 2 and 3 were significantly higher during the inner fold compared to prediction scores obtained on the outer fold, Figure 4.3. The prediction scores for the inner fold during folds 1 and 4 were also higher than the outer scores, but the points on the two graphs were more similar during these folds.

The mean test scores for the set 2 models, Figure 4.2b, show little variation between folds and very large standard deviations during all four folds. All the inner mean test prediction scores are higher than the scores obtained during the outer prediction on the test samples.

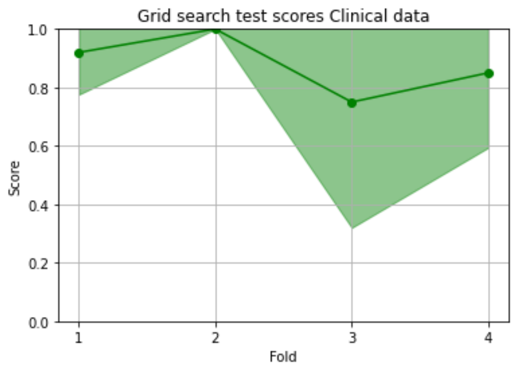
The mean test scores for the clinical data models, Figure 4.2c, show little variation between folds and extremely varying degrees of standard deviation. Fold 2 shows a perfect prediction with zero standard deviation. Again, all the mean test scores are much higher than the corresponding outer prediction scores on the test sample shown in Figure 4.4a.



(a)



(b)



(c)

Figure 4.4: Mean test scores with mean standard deviation calculated during grid search for the models selected to be in the vote classifier from each CRT subset: (a) set 1, (b) set 2, (c) clinical data.

4.3 Results from no-CRT subsets

This section presents the outer prediction scores from the majority vote classifier, as well as the inner prediction scores of the models that made up the classifier when the no-CRT subsets were fed to the work flow, as described in section 3.5.

Figure 4.5 shows the prediction scores on the test samples in each of the four outer folds described in section 3.6. The scores for set 1, set 2 and the vote classifier follow the same pattern. The score for fold 1 was high and decreased with each new fold. The score for the clinical data started high and stayed stable.

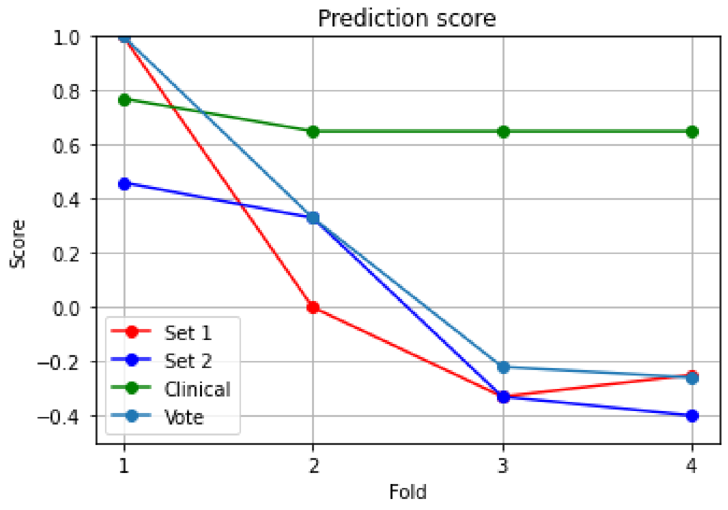


Figure 4.5: Prediction scores from the set 1, set 2 and clinical models that were chosen for the majority vote classifier and prediction scores from the majority vote classifier itself. The graphs display the MCC scores of the models across the four outer folds of the no-CRT subsets.

The mean prediction score and standard deviation for the majority vote classifier and the models that made up the classifier are listed in Table 4.4.

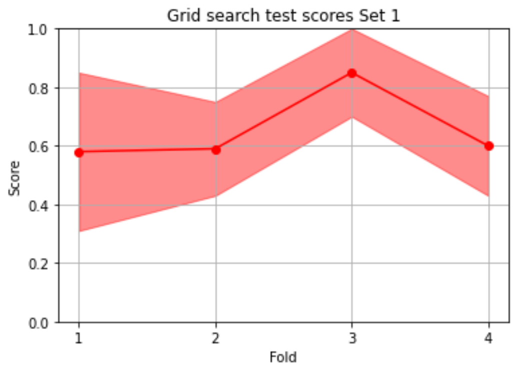
Table 4.4: The mean values and standard deviations from the outer folds of the no-CRT subset.

	Set 1	**Set 2**	**Clinical**	**Vote**
Mean	0.102	0.01	0.682	0.260
Std	0.533	0.387	0.053	0.510

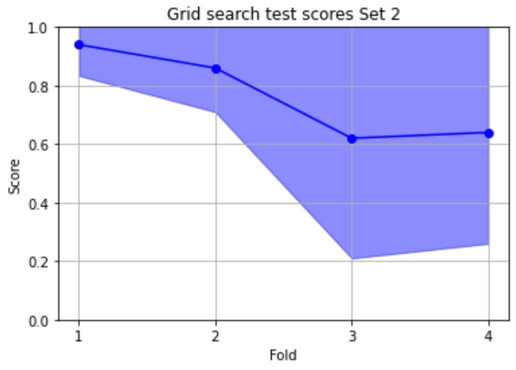
Figure 4.6 shows the mean test scores and the mean standard deviation obtained during the grid search for the models selected to be in the majority vote classifier. Three of the mean test scores from the set 1 models, Figure 4.6a, were close to 0.6, while one was slightly higher. All the inner prediction scores were significantly higher than the scores achieved during the outer predictions shown in Figure 4.5, except during fold 1.

The mean test scores from the set 2 models, Figure 4.6b, show some variation between folds and very large standard deviation during fold 3 and 4. All the inner mean test prediction scores were higher than the scores obtained from the outer predictions on the test samples.

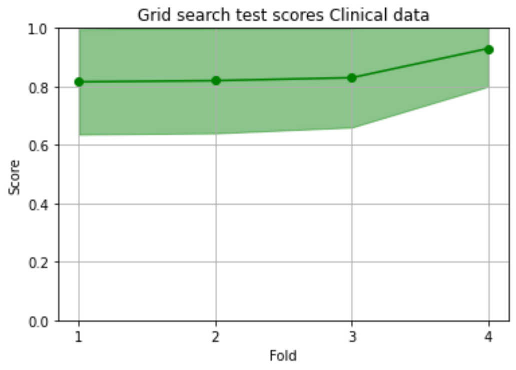
The mean test scores for the clinical data models, Figure 4.6c, were all high and showed little variation between folds. The scores are comparable to the scores achieved during the outer predictions shown in Figure 4.5.



(a)



(b)



(c)

Figure 4.6: Mean test scores with mean standard deviation calculated during grid search for the models selected to be in the vote classifier from each no-CRT subset: (a) set 1, (b) set 2, (c) clinical data.

4.4 Selected features

4.4.1 Clinical data features

Table 4.5 lists the number of features selected by RENT during each of the outer folds for the clinical data. There is not a substantial difference between the number of features selected across the folds for each subset. The CRT subset had the highest number of selected features and the no-CRT subset had the lowest.

Table 4.5: The number of features selected by RENT during the four outer folds for each subset of the clinical data.

Fold	**All-patients**	**CRT**	**no-CRT**
1	4	4	2
2	2	8	2
3	4	8	2
4	4	5	1

Table 4.6: The features selected by RENT for each of the three subsets of the clinical data. The columns show the number of folds where each specific feature was chosen by RENT.

Feature name	All	CRT	No-CRT
Gender	0	0	0
BMI	0	2	0
Age	0	1	0
Distance from anal opening	0	0	0
MSI	0	0	0
Distance from anal opening on MRI	0	0	0
mrT (TNM ed.7)	1	3	0
mrN	0	0	0
Suspicion of metastasis at time of diagnosis	4	3	4
Stage	4	4	4
Preoperative CRT	0	0	0
CEA baseline	3	4	1
CRP baseline	0	0	0
Hb baseline	0	0	0
Leukocytes	0	0	0
Sodium	0	0	0
Potassium	0	1	0
Creatinine	0	1	0
Bilirubin	0	1	0
Adjuvant treatment	2	4	1

Table 4.6 lists the features chosen by RENT for each of the three subsets of the clinical data set. The numbers corresponds to how many times a specific feature was chosen across the four outer folds. Only the feature "Stage" was selected in all four folds for every subset. Other features that were often selected were "Suspicion of metastasis at diagnosis", "CEA baseline" and "Adjuvant treatment". The CRT-subset was the subset with the most variation in selected features.

Figure 4.7 shows the τ_1 values for the features in the RENT analysis of fold 1 for the clinical all-patients subset. As listed in Table 4.1, the cutoff value for τ_1 was set to 0.2. This was a fitting setting in this case. The four prominent features met the criteria and the features that only had nonzero weights in a few models, such at the feature at index 6, were removed.

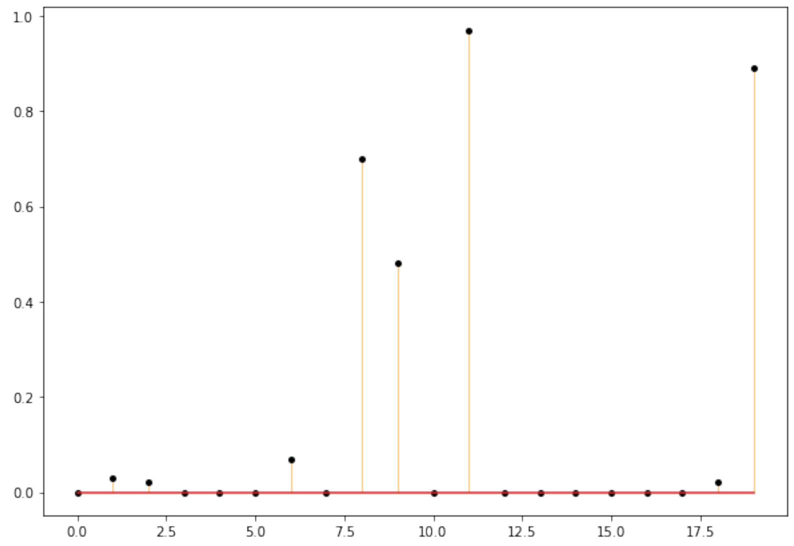


Figure 4.7: The τ_1 values from the RENT analysis of fold 1 for the clinical all-patients subset. The feature indexes are on the x-axis, and the y-axis shows the τ_1 values. The feature with the highest τ_1 value is "CEA baseline" at index 11.

4.4.2 Radiomics features

Tables 4.7 and 4.8 list the number of features selected by RENT during each of the outer folds for set 1 and set 2. The tables show substantial differences between folds for the same subset.

Table 4.7: The number of features selected by RENT during the four outer folds for each subset of set 1.

Fold	**All-patients**	**CRT**	**no-CRT**
1	75	14	186
2	77	2	51
3	20	31	19
4	2	2	41

Table 4.8: The number of features selected by RENT during the four outer folds for each subset of set 2.

Fold	**All-patients**	**CRT**	**no-CRT**
1	60	83	22
2	94	15	67
3	2	11	3
4	41	48	6

Set 1 and set 2 both contained 772 features, substantially more than the clinical data. As it would be unreasonable to list the selection frequency for all 772 features, only features that were selected by a minimum of three folds for a subset were included in Tables 4.9 and 4.10.

No features were selected during all four folds for any subset of set 1 and only four features were selected at least once by all the subsets. For set 2, 29 features were selected at least once by all subsets and four features were selected in all four folds by a single subset. The feature "glszm_GrayLevelVariance" was the most selected feature, being selected 11 out of 12 times by set 2, in additional to being selected five times by set 1.

Table 4.9: Features selected by RENT for each of the three subsets of set 1. The columns show the number of folds where each specific feature was chosen by RENT.

Feature name	**All**	**CRT**	**No-CRT**
first_order_10Percentile	3	0	3
first_order_Mean	3	0	2
first_order_Median_b6	3	2	1
first_order_RootMeanSquared_b5	3	1	0
first_order_Skewness	3	0	2
first_order_Skewness_b3	2	0	3
glcm_Autocorrelation_d_1_b3	1	0	3
glcm_DifferenceAverage_d_1	3	1	2
glcm_DifferenceEntropy_d_1	3	1	2
gldm_DependenceVariance_d_1	1	0	3
gldm_LargeDependenceHighGrayLevelEmphasis_d_1	2	0	3
gldm_LargeDependenceHighGrayLevelEmphasis_d_1_b6	3	2	0
gldm_LowGrayLevelEmphasis_d_1	3	0	3
glrlm_LongRunHighGrayLevelEmphasis	3	1	3
glrlm_LongRunHighGrayLevelEmphasis_b6	3	2	0
glrlm_LowGrayLevelRunEmphasis	3	0	3
glrlm_RunEntropy_b6	3	0	2
glrlm_ShortRunLowGrayLevelEmphasis	3	0	3
glszm_GrayLevelVariance	3	0	2
glszm_LargeAreaLowGrayLevelEmphasis_b0	0	0	3
glszm_SmallAreaHighGrayLevelEmphasis_b1	0	0	3

Table 4.10: The features selected by RENT for each of the three subsets of set 2. The columns show the number of folds where each specific feature was chosen by RENT.

Feature name	**All**	**CRT**	**No-CRT**
first_order_Kurtosis	3	1	1
first_order_Mean	3	2	2
first_order_Median_b4	3	1	1
first_order_Median_b6	3	4	1
first_order_RootMeanSquared_b5	3	3	1
first_order_Skewness	3	2	2
glcm_Autocorrelation_d_1	3	3	2
glcm_Autocorrelation_d_1_b3	1	0	3
glcm_ClusterShade_d_1_b4	3	1	2
glcm_ClusterShade_d_1_b6	2	3	1
glcm_DifferenceVariance_d_1_b0	2	4	1
glcm_Imc2_d_1	3	1	3
glcm_InverseVariance_d_1	3	1	3
glcm_InverseVariance_d_1_b4	3	1	3
glcm_JointAverage_d_1	3	2	2
glcm_JointAverage_d_1_b4	3	1	1
glcm_SumAverage_d_1	3	2	2
glcm_SumAverage_d_1_b4	3	1	1
gldm_LargeDependenceHighGrayLevelEmphasis_d_1	3	2	4
gldm_LargeDependenceHighGrayLevelEmphasis_d_1_b6	3	3	0
glrlm_LongRunHighGrayLevelEmphasis	3	2	3
glrlm_LongRunHighGrayLevelEmphasis_b5	3	1	1
glrlm_LongRunHighGrayLevelEmphasis_b6	3	3	1
glrlm_RunEntropy_b6	3	1	2
glszm_GrayLevelNonUniformity_b4	3	2	1
glszm_GrayLevelVariance	4	4	3
glszm_GrayLevelVariance_b4	2	3	0
glszm_GrayLevelVariance_b5	1	3	0
glszm_SizeZoneNonUniformityNormalized_b6	3	2	2
glszm_SmallAreaLowGrayLevelEmphasis_b1	3	3	2
glszm_ZoneEntropy_b6	3	1	3
shape_Flatness_b0	2	3	2
shape_Sphericity_b0	3	2	2

There was a large difference between the number of selected features between Fold 1 and fold 4 for the set 1 all-patients subset, see Table 4.7. Figures 4.8 and 4.9 show the τ_1 values for the features in the RENT analysis for these folds. During fold 4 the τ_1 value of 0.2 was suitable. Even though all the features had quite low τ_1 values, the two most prominent features were selected while the rest were removed. During fold 1 on the other hand, the low cutoff value resulted in an unnecessary amount of unstable features meeting the criteria and a value closer to 0.6 would have been a more suitable cutoff value.

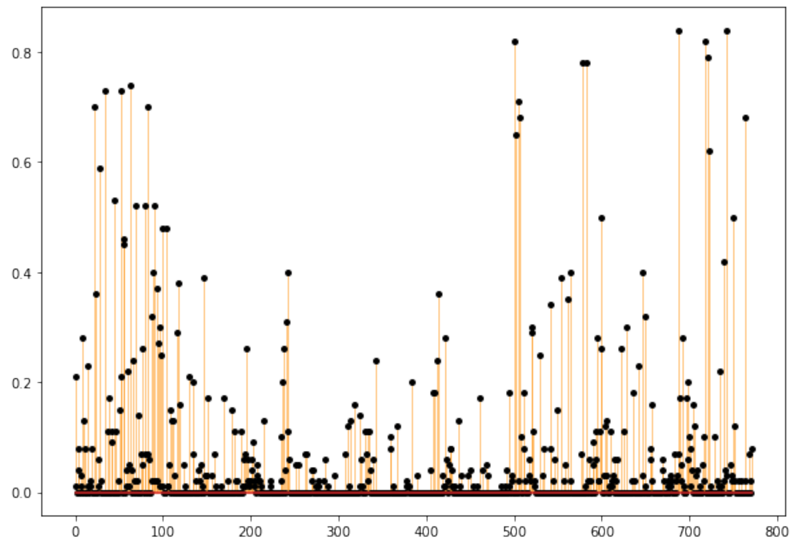


Figure 4.8: The τ_1 values from the RENT analysis of fold 1 for the clinical all-patients subset. The feature indexes are on the x-axis, and the y-axis shows the τ_1 values.

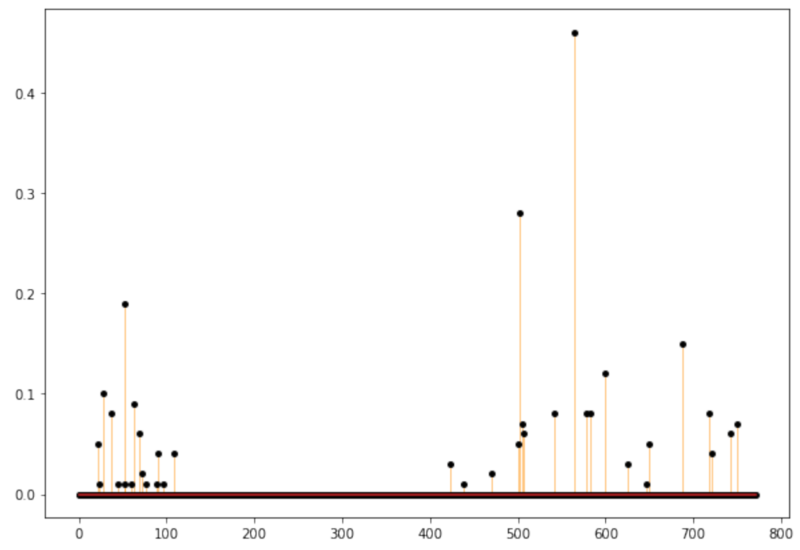


Figure 4.9: The τ_1 values from the RENT analysis of fold 4 for the set 1 all-patients subset. The feature indexes are on the x-axis, and the y-axis shows the τ_1 values.

Chapter 5

Discussion

The majority vote classifiers constructed from each subset group all had a mean score above 0 across the four outer folds. However, the relatively high mean standard deviation scores makes them an inappropriate choice for making trustworthy predictions for new data.

The models included in each of the three majority vote classifiers showed signs of overfitting, as the mean scores for the models obtained during the inner cross validation was higher than the mean scores obtained during the outer cross validation.

5.1 Interpreting results

The purpose behind ensemble learning is to improve performance and robustness. The figures with predictions from the outer folds, Figures 4.1, 4.3 and 4.5, do not show any signs of the majority vote classifier performing better than its top-performing model.

There were big differences in the predictive scores of the majority vote classifiers across the outer folds. This indicates that the classifiers are sensitive to how the datasets were split for the outer cross validation. The sensitivity is more prominent for the CRT and no-CRT subsets compared to the all-patients subsets, most likely because of the reduced number of patients in these two subsets.

The models based on the clinical data had the highest mean prediction score and the lowest standard deviation across the outer folds for the all-patients and no-CRT subsets. This is likely because the feature selection process for this dataset was more stable compared to the set 1 and set 2 datasets. The clinical dataset only contained 20 features and the same features were often chosen in multiple folds, even across its three subsets. The subset where the clinical data showed the most variation across the outer folds was the CRT subset. This was also the clinical data subset with the least stable feature selections.

The models based on set 1 had the highest mean prediction scores and the lowest standard deviation across the outer fold for the CRT subsets.

5.2 Evaluation process

The purpose behind using k-fold CV for evaluation is to acquire better estimates of the true predictive power of a model. The big differences in prediction scores between folds indicate that the predictive performance of the majority vote classifier is entirely dependent on the train-test split. Performing repeated k-fold CV for the outer cross validation could have mitigated some of these differences.

Repeated k-fold CV repeats the cross validation process multiple times by reshuffling the dataset into new train-test splits [18]. By performing repeated k-fold CV, the influence on the mean prediction scores from unlucky splits would be reduced. This would lead to more robust results and allow for a more accurate interpretation of those results.

5.3 Selected features

The features chosen by RENT for the clinical data were all features known to hold medical predictive power. Take the feature "Stage" which is directly proportional to the progression of a cancer. A high cancer stage would mean that the cancer has spread to nearby tissue or other parts of the body, which would mean that more invasive surgery and other treatments mentioned in section 2.1.3 might be needed. All of these treatment options come with side effects that can greatly affect a patient's overall health and introduce more risks [30] [31].

Looking at which features were not chosen could also be of interest. High age and BMI are both factors that increase the risk of cancer, but they do not hold much predictive power for whether or not a patient survives, in this particular data.

Since the features chosen by RENT for the clinical datasets were all features with known medical predictive power, the analysis does not add much of value to medical literature, apart from indicating which features play the most important roles when predicting treatment outcome. It could be interesting to see how well the models would perform if the most predictive features were removed. In this situation the models would be forced to rely on other information.

Set 1 and set 2 both contain a total of 772 features. Out of these, 21 and 33 features were frequently chosen by RENT and included in Tables 4.9 and 4.10. The tables only include features that were chosen during at least three out of four folds for a subset of set 1 or set 2. This means that a feature may have been chosen during two out of four folds for all the subsets and still not have been included. The criteria for being included in either table derby depend more on internal feature stability for one subset than feature stability across all subsets. Some features, such as "glszm_LargeAreaLow GrayLevelEmphasis_b0" which was chosen during three out of four folds for one subset, but was never chosen by any of the other subsets, might not have been included if the criteria was more focused on stability across subsets.

It should also be noted that there was a big difference in the number of features chosen by RENT between folds. For the set 2 all-patients subset, 94 features were chosen during fold 2 while only two were chosen during fold 3. This shows how much feature selection varies based on which samples end up in which fold. By performing repeated k-fold CV this problem could be somewhat mitigated.

Greater stability across the folds might be achieved by first selecting features that are prominent and then using a new dataset that only includes those features. This way a lot of noise is removed, which should lead to more stable models [32].

5.4 Robustness of radiomic features

The biggest challenge in radiomics is ensuring reliable and reproducible output. Research has shown that specific study conditions and authors' choices have great influence over the results [27].

Voxel intensities in MRIs do not have fixed tissue-specific values. This means that grey-level intensity may change between MRI sessions for the same patient, even if the patient is scanned in the same position by the same scanner using the same RF sequence. However, tissue contrast remains the same [**radi˙fact˙and˙challenges**].

One possibility is to focus on the texture features. Texture features are dependent on the relationship between voxels and not on the numeric values of the grey-level intensities themselves. However, texture features have shown sensitivity to variations in acquisition parameters, such as spatial resolution. Another possibility is to normalise the images before feature extraction [**radi˙fact˙and˙challenges**].

5.5 Selected models and data sets

Across all the results shown in Sections 4.1, 4.2 and 4.3, a total of 36 models were chosen to be part of the majority vote classifier. Out of these 36 models, as many as 30 were based on RENT, while three were based on the full data set and three were based on PCA. Out of the different machine learning algorithms, KNN models were selected five times, logistic regression models eight times, random forest models seven times and SVC models 16 times. All the KNN models were selected by clinical data models and half of the SVC models were selected by set 2 models.

5.6 Choice of response variable

The data included several possible response variables and groupings. The response variables were binary events such as progression free survival (PFS) and overall survival (OS). PFS refers to whether a patient experienced any form of cancer recurrence before the end of the study, while OS refers to whether or not a patient is alive at the end of the study. OS is not concerned with cancer recurrence. The data also included the length of time between a patient joining the study and the PFS-event or OS-event occurring. Other possible response variables could have been survival after three years or metastasis after three years.

In this thesis, OS was chosen as the response variable. As mentioned in section 3.3, four patients were removed because the listed cause of death was different from colorectal cancer. Most patients that had passed away did not have a listed cause of death. This opens up the possibility that more patients died of other causes. However, if that is the case, cancer or cancer treatment could still have played a role in these deaths. Both cancer and cancer treatment can reduce a patient's overall health and lead to increased risk of death from other diseases.

For patients that received radiotherapy, it could be interesting to predict how well they responded to the therapy. Response can be measured by the tumor regression grade (TRG), where 1-2 is bad and 3-4 is good.

5.7 Parameter choices

In order to be able to compare the different models, the same RENT and PCA parameters had to be used for all datasets in the same block. For RENT this meant that τ_1, τ_2 and τ_3 could not be changed to account for differences between folds. For simplicity's sake, the parameter values were set to the same value for all the datasets.

Depending on the dataset that was being treated, the number of features selected by RENT varied greatly. During some outer folds the number of selected features for set 1 and set 2 exceeded fifty, while during other folds the number was as low as two. The number of selected features for set 1 no-CRT during fold 1 even reached 186.

During some folds, RENT would not select any feature unless the τ_1, τ_2 and τ_3 values were set quite low. During other folds, the same low parameter values resulted in RENT selecting a number of unstable features. Using higher parameter values during these folds would have been a way to avoid selecting the more unstable features. This accounts for the huge difference in the number of selected features across folds. Altogether, this shows that many of the features selected by RENT were not stable.

While the features included in Tables 4.9 and 4.10 were chosen quite often for a single subset, many were not selected universally across all the subsets like the most selected features in Table 3.2. This also supports the claim that most features selected by RENT were unstable.

In PCA all datasets from the same block had to be reduced to the same number of principal components. The number of principal components was set to 10 for all datasets without exploring differences in explained variance. This could be the reason why datasets treated by PCA were infrequently chosen for the majority vote classifier.

5.8 Personalised medicine

Cancer treatment is by an large adjusted to suit individual patients [10], but treatment is also based on experiences and knowledge built across years of research and on what works on average.

When it comes to cancer treatment, patients need to be considered individually in relation to genetics, environment and other underlying medical conditions or complications. All in all, medical trials contain very heterogeneous data. For the purpose of this thesis, more attention to smaller groupings within the data and larger selection of patients could have helped in bringing forth better results. This was the reasoning behind dividing the datasets into three subsets based CRT treatment. Patients who undergo CRT treatment typically have locally advanced cancers: cancer at stage T4 or T3. This subset of patients may be a more homogeneous group in comparison to the whole patient cohort.

5.9 Future work

The main concern in this thesis' results was the lack of stability in the radiomics features. Further analysing which features were the most stable and only including them in an analysis would make the models more similar and might bring better results. Performing repeated k-fold CV on the outer folds would mitigate some of the affects from unlucky train-test splits and enable a more accurate estimate of the prediction score of the workflow.

Further exploring the impact of different parameter settings in RENT could lead to more stable feature selections. Since the explained variance in the principal components were not explored, it is possible that better performance could be achieved by reducing the number of components based on each data block. Limiting the number of principal components could remove more noise.

Other ensemble methods could be explored, for example dynamic ensemble selection. The majority vote classifier ensemble method is a static ensemble method where the same models are used to predict all test samples. Dynamic ensemble selection involves selecting models from the ensemble that are expected to perform well on individual samples [33]. Dynamic ensemble selection can be implemented with DESlib, whose documentation can be found at [34].

Chapter 6

Conclusion

The majority vote classifiers that were constructed to predict treatment outcome were sub-optimal. In most instances, simply using the models based on information from the clinical data, while disregarding the models based on radiomics, would have resulted in better predictions. A likely reason for this is that the radiomics features were unstable.

The RENT analysis on the radiomics features showed some promise. Out of the original 772 features, 21 and 33 features were selected regularly, depending on the voxel resolution of the MRI images the information was extracted from.

This could indicate that if features with predictive power are present in the radiomics data, both RENT and PCA are unable to fully capture them or the models are unable to fully utilise the features. With more time, the impact of different parameter settings could have been explored, for instance in relation to feature selection and model stability.

The instability of the current models makes them an inappropriate source for trustworthy predictions on new data. However, in the future, machine learning could give valuable information concerning cancer treatment and prognosis, given more data and better data quality, in addition to further exploration of different analysis methods.

Bibliography

[1] World Health Organization. *Cancer*. 2021. URL: https://www.who.int/news-room/fact-sheets/detail/cancer.

[2] Hyuna Sung. *Global cancer statistics 2020: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries*. 2021. URL: https://acsjournals.onlinelibrary.wiley.com/doi/10.3322/caac.21660.

[3] American Cancer Institute. *MRI for Cancer*. URL: https://www.cancer.org/treatment/understanding-your-diagnosis/tests/mri-for-cancer.html.

[4] Robert J. Gillies, Paul E. Kinahan, and Hedvig Hricak. *Radiomics: Images Are More than Pictures, They Are Data*. 2015. URL: https://pubs.rsna.org/doi/10.1148/radiol.2015151169.

[5] Kreftforeningen. *Hva er kreft?* 2020. URL: https://kreftforeningen.no/om-kreft/hva-er-kreft/.

[6] Cancer Research UK. *Bowel cancer*. 2018. URL: https://www.cancerresearchuk.org/about-cancer/bowel-cancer.

[7] Kreftforeningen. *Tarmkreft*. 2021. URL: https://kreftforeningen.no/om-kreft/kreftformer/tarmkreft/.

[8] Cancer Research UK. *Risks and causes*. 2018. URL: https://about-cancer.cancerresearchuk.org/about-cancer/bowel-cancer/risks-causes.

[9] National Cancer Institute. *Cancer Staging*. 2015. URL: https://www.cancer.gov/about-cancer/diagnosis-staging/staging.

[10] Arild Nesbakken, Stein G. Larsen, and Marianne Grønlie Guren. *Treatment of cancer in the colon and rectum*. 2015. URL: http://oncolex.org/Oncolex/Colorectal-cancer/Procedures/TREATMENT.

[11] Cancer Research UK. *Treatment decisions for rectal cancer*. 1018. URL: https://www.cancerresearchuk.org/about-cancer/bowel-cancer/treatment/treatment-rectal/treatment-decisions.

[12] National Cancer Institute. *Radiation Therapy to Treat Cancer*. 2019. URL: https://www.cancer.gov/about-cancer/treatment/types/radiation-therapy.

[13] National Cancer Institute. *How Is Chemotherapy Used to Treat Cancer?* 2019. URL: https://www.cancer.org/treatment/treatments-and-side-effects/treatment-types/chemotherapy/how-is-chemotherapy-used-to-treat-cancer.html.

[14] Donald W. McRobbie et al. *MRI From Picture to Proton*. second edition. 2006.

[15] Robert A. Pooley. "Fundamental Physics of MR Imaging". In: *RadioGraphics* 25.4 (2005). PMID: 16009826, pp. 1087–1099. DOI: 10.1148/rg.254055027. eprint: https://doi.org/10.1148/rg.254055027. URL: https://doi.org/10.1148/rg.254055027.

[16] Jan A. Holtet. *Jordmagnetisme*. 2021. URL: https://snl.no/jordmagnetisme.

[17] Bachir Taouli and Dow-Mu Koh. *Diffusion-weighted MR Imaging of the Liver*. 2009. URL: https://pubs.rsna.org/doi/10.1148/radiol.09090021.

[18] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning*. second edition. 2018.

[19] Anna Jenul et al. *RENT – Repeated Elastic Net Technique for Feature Selection*. 2021. arXiv: 2009.12780 [cs.LG].

[20] Scikit-learn developers. *Toy datasets*. URL: https://scikit-learn.org/stable/datasets/toy_dataset.html.

[21] Scikit-learn developers. *1.4. Support Vector Machines*. URL: https://scikit-learn.org/stable/modules/svm.html.

[22] Scikit-learn developers. *sklearn.metrics.matthews*$_c$*orrcoef*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html#sklearn.metrics.matthews_corrcoef.

[23] Kathrine Røe Redalen. *Functional MRI of Hypoxia-mediated Rectal Cancer Aggressiveness (OxyTarget)*. 2020. URL: https://clinicaltrials.gov/ct2/show/NCT01816607.

[24] *The OxyTarget study*. URL: https://www.acredit.no/the-oxytarget-study/.

[25] AAse Langan. "MRI-Based Radiomics Analysis for Predicting Treatment Outcome in Rectal Cancer". MA thesis. NTNU, 2020.

[26] Bakke KM, Grøvik E, and Meltzer S et al. *Comparison of Intravoxel incoherent motion imaging and multiecho dynamic contrast-based MRI in rectal cancer. J Magn Reson Imaging*. 2019. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/jmri.26740.

[27] Janita E. van Timmeren et al. *Radiomics in medical imaging—"how-to" guide and critical reflection*. 2020. URL: https://insightsimaging.springeropen.com/articles/10.1186/s13244-020-00887-2#Abs1.

[28] Pyradiomics community. *Welcome to pyradiomics documentation!* 2016. URL: https://pyradiomics.readthedocs.io/en/latest/index.html.

[29] Ahmed Albuni. *Biorad*. 2020. URL: https://github.com/ahmedalbuni/biorad.

[30] American Cancer Society. *Chemotherapy Side Effects*. 2020. URL: https://www.cancer.org/treatment/treatments-and-side-effects/treatment-types/chemotherapy/chemotherapy-side-effects.html.

[31] National Cancer Institute. *Radiation Therapy Side Effects*. 2018. URL: https://www.cancer.gov/about-cancer/treatment/types/radiation-therapy/side-effects.

[32] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. 3rd edition. 2000.

[33] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. 1st edition. 2021.

[34] Rafael M. O. Cruz et al. *Welcome to DESlib documentation!* 2020. URL: https://deslib.readthedocs.io/en/latest/index.html.

Appendix A

Models

A.1 Models from the all-patients subsets

Table A.1: Models selected from fold 1 with the all-patients subsets.

	Set 1	Set 2	Clinical
Data	Untreated	RENT	RENT
Model Type	Random forest classifier	SVC	SVC
Parameters	criterion: 'gini' max_depth: 3 n_estimators: 1	C: 4 class_weight: 'None' kernel: 'sigmoid'	C: 1 class_weight: 'balanced' svc_kernel: 'sigmoid'
Mean Test Score	0.433	0.600	0.710
Mean Train Score	0.501	0.576	0.664
Std Test Score	0.210	0.180	0.300
Std Train Score	0.088	0.096	0.102

Table A.2: Outer fold prediction from fold 1 with the all-patients subsets.

Set	Score
Set 1	0.0
Set 2	0.533
Clinical	0.756
Voter	0.533

Table A.3: Models selected from fold 2 with the all-patients subsets.

	Set 1	Set 2	Clinical
Data	RENT	RENT	PCA
Model Type	SVC	SVC	Random forest classifier
Parameters	C: 1 class_weight: 'balanced' kernel: 'sigmoid'	C: 4 class_weight: 'None' kernel: 'sigmoid'	criterion: 'gini' weights: 'uniform' n_estimators: 21
Mean Test Score	0.500	0.566	0.715
Mean Train Score	0.987	0.560	1.0
Std Test Score	0.231	0.078	0.116
Std Train Score	0.023	0.098	0.0

Table A.4: Outer fold prediction from fold 2 with the all-patients subsets.

Set	Score
Set 1	0.1
Set 2	0.213
Clinical	0.554
Voter	0.354

Table A.5: Models selected from fold 3 with the all-patients subsets.

	Set 1	Set 2	Clinical
Data	RENT	RENT	RENT
Model Type	Logistic regression	SVC	k neighbors classifier
Parameters	C: 1 penalty: ’l2’	C: 1 class_weight: ’balanced’ kernel: ’linear’	n_neighbors: 3 weights: ’uniform’
Mean Test Score	0.520	0.530	0.801
Mean Train Score	0.657	0.520	0.810
Std Test Score	0.144	0.145	0.140
Std Train Score	0.093	0.062	0.040

Table A.6: Outer fold prediction from fold 3 with the all-patients subsets.

Set	Score
Set 1	0.440
Set 2	0.0
Clinical	0.3
Voter	0.3

Table A.7: Models selected from fold 4 with the all-patients subsets.

	Set 1	Set 2	Clinical
Data	RENT	RENT	RENT
Model Type	Random forest classifier	SVC	SVC
Parameters	criterion: ’entropy’ max_depth: 1 n_estimators: 41	C: 8 class_weight: ’balanced’ kernel: ’rbf’	C: 2 class_weight: ’balanced’ kernel: ’poly’
Mean Test Score	0.510	0.634	0.611
Mean Train Score	0.540	1	0.641
Std Test Score	0.103	0.253	0.283
Std Train Score	0.075	0	0.100

Table A.8: Outer fold prediction from fold 4 with the all-patients subsets.

Set	Score
Set 1	0.0
Set 2	-0.258
Clinical	0.440
Voter	0.0

A.2 Models from the CRT subsets

Table A.9: Models selected from fold 1 with the CRT subsets.

	Set 1	Set 2	Clinical
Data	RENT	RENT	RENT
Model Type	SVC	SVC	Kneighbors classifier
Parameters	C: 8 class_weight: 'balanced' kernel: 'sigmoid'	C: 1 class_weight: 'balanced' kernel': 'sigmoid'	n_neighbors: 6 weights: 'distance'
Mean Test Score	0.820	0.758	0.917
Mean Train Score	0.752	0.853	1
Std Test Score	0.192	0.286	0.144
Std Train Score	0.111	0.102	0

Table A.10: Outer fold prediction from fold 1 with the CRT subsets.

Set	Score
Set 1	0.645
Set 2	0.548
Clinical	0.0
Voter	0.645

Table A.11: Models selected from fold 2 with the CRT subsets.

	Set 1	Set 2	Clinical
Data	RENT	RENT	PCA
Model Type	SVC	Random forest classifier	Logistic regression
Parameters	C: 8 class_weight: None kernel: 'linear'	criterion: 'entropy' max_depth: 4 n_estimators: 21	C: 1 penalty: 'none'
Mean Test Score	0.545	0.635	1
Mean Train Score	0.551	0.970	1
Std Test Score	0.358	0.265	0
Std Train Score	0.111	0.051	0

Table A.12: Outer fold prediction from fold 2 with the CRT subsets.

Set	Score
Set 1	0.0
Set 2	-0.4
Clinical	-0.09
Voter	-0.258

Table A.13: Models selected from fold 3 with the CRT subsets.

	Set 1	Set 2	Clinical
Data	RENT	RENT	RENT
Model Type	SVC	Logistic regression	Logistic regression
Parameters	C: 6 class_weight: 'balanced' kernel: 'sigmoid'	C: 1 penalty: 'none'	C: 1 penalty: 'l2'
Mean Test Score	0.593	0.573	0.75
Mean Train Score	0.617	1	1
Std Test Score	0.106	0.361	0.433
Std Train Score	0.120	0	0

Table A.14: Outer fold prediction from fold 3 with the CRT subsets.

Set	Score
Set 1	0.548
Set 2	0.730
Clinical	0.417
Voter	0.730

Table A.15: Models selected from fold 4 with the CRT subsets.

	Set 1	Set 2	Clinical
Data	PCA	RENT	RENT
Model Type	Random forest classifier	SVC	SVC
Parameters	criterion: 'gini' max_depth: 4 n_estimators: 1	C: 8 class_weight: 'balanced' kernel: 'sigmoid'	C: 2 class_weight: 'balanced' kernel: 'sigmoid'
Mean Test Score	0.710	0.712	0.852
Mean Train Score	0.707	0.80	0.805
Std Test Score	0.172	0.301	0.256
Std Train Score	0.103	0.126	0.156

Table A.16: Outer fold prediction from fold 4 with the CRT subsets.

Set	Score
Set 1	0.25
Set 2	0.25
Clinical	-0.5
Voter	0.0

A.3 Models from the no-CRT subsets

Table A.17: Models selected from fold 1 with the no-CRT subsets.

	Set 1	Set 2	Clinical
Data	RENT	RENT	RENT
Model Type	Logistic regression	Logistic regression	K neighbors classifier
Parameters	C: 1 penalty: 'l2'	C: 1 penalty: 'l2'	n_neighbors: 1 weights: 'uniform'
Mean Test Score	0.582	0.940	0.816
Mean Train Score	1	1	0.780
Std Test Score	0.274	0.106	0.183
Std Train Score	0	0	0.058

Table A.18: Outer fold prediction from fold 1 with the no-CRT subsets.

Set	Score
Set 1	1.0
Set 2	0.467
Clinical	0.774
Voter	1.0

Table A.19: Models selected from fold 2 with the no-CRT subsets.

	Set 1	Set 2	Clinical
Data	RENT	RENT	ALL
Model Type	SVC	SVC	Random forest classifier
Parameters	C: 4 class_weight: 'balanced' kernel: 'sigmoid'	C: 3 class_weight: 'balanced' kernel: 'sigmoid'	criterion: 'entropy' max_depth: 3 n_estimators: 61
Mean Test Score	0.590	0.861	
Mean Train Score	0.550	0.920	
Std Test Score	0.165	0.147	
Std Train Score	0.030	0.0556	

Table A.20: Outer fold prediction from fold 2 with the no-CRT subsets.

Set	Score
Set 1	0.0
Set 2	0.333
Clinical	0.655
Voter	0.333

Table A.21: Models selected from fold 3 with the no-CRT subsets.

	Set 1	Set 2	Clinical
Data	RENT	RENT	RENT
Model Type	Logistic regression	Logistic regression	K neighbors classifier
Parameters	C: 1 penalty: 'l2'	C: 1 penalty: 'l2'	n_neighbors: 3 weights: 'uniform'
Mean Test Score	0.853	0.625	0.835
Mean Train Score	0.953	0.708	0.800
Std Test Score	0.152	0.415	0.167
Std Train Score	0.047	0.106	0.068

Table A.22: Outer fold prediction from fold 3 with the no-CRT subsets.

Set	Score
Set 1	-0.333
Set 2	-0.333
Clinical	0.655
Voter	-0.218

Table A.23: Models selected from fold 4 with the no-CRT subsets.

	Set 1	Set 2	Clinical
Data	RENT	RENT	All
Model Type	Random forest classifier	SVC	K neighbors classifier
Parameters	criterion: 'gini' max_depth: 2 n_estimators: 21	C: 8 class_weight: 'balanced kernel: 'linear'	n_neighbors: 1 weights: 'uniform'
Mean Test Score	0.592	0.645	0.923
Mean Train Score	0.721	0.917	1
Std Test Score	0.171	0.382	0.127
Std Train Score	0.069	0.055	0

Table A.24: Outer fold prediction from fold 4 with the no-CRT subsets.

Set	Score
Set 1	-0.258
Set 2	-0.4
Clinical	0.645
Voter	-0.258

Appendix B

Python code

B.1 Machine learning algorithms

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 22 21:35:11 2021

@author: Marthe Søvdsnes
"""

from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import matthews_corrcoef
from sklearn.metrics import make_scorer
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings("ignore")

def KNN(x_train, y_train):
    """
    Trains a K nearest neighbour model including grid search.
    Prints the best mean test score and the best parameters

    Parameters
    ----------
    x_train: df
        the training data
    x_test: df
        The test data

    Returns
    ---------
    model
        object
    """

    param_grid = {'kneighborsclassifier__n_neighbors': [x for x in range(1, 10, 1)],
                  'kneighborsclassifier__weights': ['uniform', 'distance']}

    pipe_k = make_pipeline(StandardScaler(), KNeighborsClassifier(n_jobs=-1))
    gs = GridSearchCV(estimator=pipe_k,
                      scoring=make_scorer(matthews_corrcoef),
                      param_grid=param_grid,
                      cv=4,
                      return_train_score=True)

    model = gs.fit(x_train, y_train)
    print('Test score KNN:', gs.best_score_)
    print(gs.best_params_)

    return model

def RF(x_train, y_train):
    """
    Trains a Random Forest model including grid search.
    Prints the best mean test score and the best parameters

    Parameters
    ----------
    x_train: df
```

```
        the training data
    x_test: df
        The test data

    Returns
    ---------
    model
        object
    """

    param_grid = {'randomforestclassifier__n_estimators': [x for x in range(1, 100, 20)],
                  'randomforestclassifier__criterion': ['gini', 'entropy'],
                  'randomforestclassifier__max_depth': [x for x in range(1, 10, 1)]}

    pipe_rf = make_pipeline(RandomForestClassifier())

    gs = GridSearchCV(estimator=pipe_rf,
                      scoring=make_scorer(matthews_corrcoef),
                      param_grid=param_grid,
                      cv=4,
                      return_train_score=True)

    model = gs.fit(x_train, y_train)
    print('Test score Random Forest:', gs.best_score_)
    print(gs.best_params_)

    return model

def sv(x_train, y_train):
    """
    Trains a support vector classifier model including grid search.
    Prints the best mean test score and the best parameters

    Parameters
    ----------
    x_train: df
        the training data
    x_test: df
        The test data

    Returns
    ---------
    model
        object
    """

    param_grid = {'svc__C': [x for x in range(1, 10, 1)],
                  'svc__kernel': ['linear', 'poly', 'rbf', 'sigmoid', 'precomputed'],
                  'svc__class_weight': ['balanced', None]}

    pipe_svc = make_pipeline(StandardScaler(), svm.SVC())

    gs = GridSearchCV(estimator=pipe_svc,
                      scoring=make_scorer(matthews_corrcoef),
                      param_grid=param_grid,
                      cv=4,
                      return_train_score=True)

    model = gs.fit(x_train, y_train)
    print('Training score Support Vector:', gs.best_score_)
    print(gs.best_params_)
    return model

def LR(x_train, y_train):
    """
    Trains a Logistic regression model including grid search.
    Prints the best mean test score and the best parameters

    Parameters
    ----------
    x_train: df
        the training data
    x_test: df
        The test data

    Returns
    ---------
    model
        object
    """

    param_grid = {'logisticregression__C': [x for x in range(1, 10, 1)],
                  'logisticregression__penalty': ['l1', 'l2', 'elasticnet', 'none']}

    pipe_lr = make_pipeline(StandardScaler(), LogisticRegression())

    gs = GridSearchCV(estimator=pipe_lr,
                      scoring=make_scorer(matthews_corrcoef),
                      param_grid=param_grid,
                      cv=4,
                      return_train_score=True)

    model = gs.fit(x_train, y_train)
    print('Training score Logistic Regression:', gs.best_score_)
    print(gs.best_params_)

    return model
```

B.2 Data preprocessing

```python
# -*- coding: utf-8 -*-
"""
Created on Wed Mar 24 12:54:28 2021

@author: Marthe Søvdsnes
"""

import numpy as np
import pandas as pd
import re

def targets(df, crt=False):
    """
    Attaches the targets from the clinical data to the MRI data sets
    and removes patients with other causes of death.

    Parameters
    ----------
    df: DataFrame
        MRI data set without targets
    crt: bool, optional
        decides whether the data should be split based on the crt column.
        Default is False, no split.

    Returns
    ----------
    DataFrame
        input dataframe with targets attached. If crt=False returns
        one dataframe
        Returns two dataframes if crt=True where first has crt="Nei"
        and second has crt="Ja". Nei=No, Ja=Yes.
    """

    medical = pd.read_excel("clinical_data.xlsx")

    i = 0
    for r in df.iloc[:, 0]: # changes the string format of the 'Name' column
                            # in the MRI data so that it matches with clinical data
        number = [int(s) for s in re.findall(r'\d+', r)][0]
        if number < 10:
            df.iloc[i, 0] = 'OxyTarget 00{}'.format(number)
        elif number < 100:
            df.iloc[i, 0] = 'OxyTarget 0{}'.format(number)
        else:
            df.iloc[i, 0] = 'OxyTarget {}'.format(number)
        i = i + 1

    df = drop_other_death_causes(df) # this is here because the potential split
                                     # that occurs under
    if not crt: # Default, if you don't want to divide the set based on crt
        medical = medical[['ID', 'OS-event', 'Tid til OS-event', 'PFS-event',
                           'Tid til PFS-event']]
        return pd.merge(df, medical, how='left', left_on='Name',
                        right_on='ID').drop(['ID', 'Name'], axis=1)

    else: #divides the set based on the crt column. NB!! returns two dataframes
        medical = medical[['ID', 'Preoperativ CRT     (Ja/Nei)', 'OS-event',
                           'Tid til OS-event', 'PFS-event', 'Tid til PFS-event']]
        df = pd.merge(df, medical, how='left', left_on='Name', right_on='ID')

        df1 = df[df['Preoperativ CRT     (Ja/Nei)'] == 'Nei']
        df1 = df1.drop(['ID','Name', 'Preoperativ CRT     (Ja/Nei)'], axis=1)

        df2 = df[df['Preoperativ CRT     (Ja/Nei)'] == 'Ja']
        df2 = df2.drop(['ID', 'Name', 'Preoperativ CRT     (Ja/Nei)'], axis=1)
        return df1, df2

def drop_other_death_causes(df):
    """
    Removes the patients with cause of death unrelated to colorectal cancer

    Parameters
    ----------
    df: DataFrame

    Returns
    ----------
    dataframe
    """

    df = df[~df['Name'].isin(['OxyTarget 138', 'OxyTarget 153',
                              'OxyTarget 176', 'OxyTarget 029'])]
    return df

def MRI_data(fileloc, crt=False):
    """
    Imports and changes the MRI data into desired format

    Parameters
    ----------
    fileloc: str
        location of MRI file

    Returns
    ---------
    DataFrame
    """

    df = pd.read_csv(fileloc)
    df = targets(df, crt)
    return df

def load_clinical_data():
    """
    Reads the clinical data, removes patients with casue of
    death unrelated to colorectal cancer and unneccesary columns

    Parameters
    ----------
    None

    Returns
    ---------
    dataframe
            dataframe of clinical data with some rows and columns removed
    """

    clin = pd.read_excel("clinical_data.xlsx")
    filelocation = "Set2_Combined_v2.csv" # MRI file
    df = pd.read_csv(filelocation)

    i = 0
    for r in df.iloc[:, 0]:
        number = [int(s) for s in re.findall(r'\d+', r)][0]
        if number < 10:
            df.iloc[i, 0] = 'OxyTarget 00{}'.format(number)
        elif number < 100:
            df.iloc[i, 0] = 'OxyTarget 0{}'.format(number)
        else:
            df.iloc[i, 0] = 'OxyTarget {}'.format(number)
        i = i + 1

    df = df[['Name']]
    clin = pd.merge(df, clin, how='left', left_on='Name', right_on='ID').drop(['ID'], axis=1)
    clin = drop_other_death_causes(clin).drop(['Name'], axis=1)
    clin = clin.reset_index().drop(['index'], axis=1)
    columns = ['Inklusjonsdato', 'Blodprøver ved inklusjon', 'Høyde    (cm)',
               'Vekt  (kg)', 'Symptomer', 'Dato henvist til spesialist',
               'Lokalisasjon primærtumor', 'Dato colono-/rektoskopier m/biopsi',
               'Histologi av biopsi preparatnr.', 'Cancer klassif./type ',
               'Dato MR m/diagnose', 'Lokalisasjon metastaser',
               'Dato/type annen radiologi/intervensjon ved diagnosetidspunkt',
               'Dato oppstart preoperativ kjemoterapi',
               'Dato preoperativ radioterapi', 'Type preoperativ radioterapi',
               'Dato avslutt preoperativ radioterapi',
               'Dato MR kontroll etter preoperativ CRT',
               'Beskrivelse av MR etter preoperativ CRT',
               'Dato kirurgi', 'Sted kirurgi',
               'Type kirurgi                    (opersjonsbeskrivelse)',
               'Histologi (kirurgi) preparatnr.', 'Histologisk beskrivelse',
               'Kommentar patologi', 'Andre undersøkelser / radiologi etc',
               'Blodtype', 'Kommentar adjuvant behandling', 'Dato metastaser',
               'Type metastase', 'Dato lokalt recidiv', 'Type lokalt recidiv',
               'MORS', 'Annen kreftsykdom', 'Inkludert i andre studier',
               'Siste reg OS', 'Planlagt oppfølging', 'Kommentar',
               'Antall lymfeknuter undersøkt', 'Differensiering',
               'Type preoperativ radioterapi',  'Type preoperativ kjemoterapi']
    return clin.drop(columns, axis=1)

def clean_data(df):
    """
    Cleans the data and makes it usable by machine learning algorithms

    Parameters
    ----------
    df: dataframe

    Returns
    ----------
    dataframe
    """

    df['Kjønn     (K/M)'] = np.where(df['Kjønn     (K/M)']=='M', 1, 0)
    for col in df.columns:
        df[col] = df[col].apply(lambda x: 1 if x == 'Positiv'  else x)
        df[col] = df[col].apply(lambda x: 0 if x == 'Negativ'  else x)
        df[col] = df[col].apply(lambda x: 1 if x == 'Ja'  else x)
        df[col] = df[col].apply(lambda x: 1 if x == 'ja'  else x)
        df[col] = df[col].apply(lambda x: 0 if x == 'Nei'  else x)
        df[col] = df[col].apply(lambda x: np.nan if x == '-'  else x)
        df[col] = df[col].apply(lambda x: np.nan if x == 'Missing'  else x)
        df[col] = df[col].apply(lambda x: np.nan if x == 'Pending'  else x)
        df[col] = df[col].apply(lambda x: np.nan if x == 'NF'  else x)
        df[col] = df[col].apply(lambda x: 13.9 if x == '13,9'  else x)
        df[col] = df[col].apply(lambda x: 4 if x == '4a'  else x)
    return df

def remove_nan_row(df, k, crt=False):
    """
    Remove columns with nan > k, then removes all rows that still contains nan.
    Prints shape of returned dataframe. Prints procentage of data that is class 1.

    Parameters
    ----------
    df: dataframe
    k: int
        the higest number of nan a coumns can have and not be removed

    Returns
    ----------
    dataframe
    """

    f = df.isna().sum()
    too_many_nan = []
    index = 0
    for i in f:
        if i > k:
            too_many_nan.append(f.index[index])
        index = index + 1

    df = df.drop(too_many_nan, axis=1)
    df = df.dropna(axis=0)

    if crt: # Default, if you don't want to divide the set based on crt
        df1 = df[df['Preoperativ CRT     (Ja/Nei)'] == 0]
        df2 = df[df['Preoperativ CRT     (Ja/Nei)'] == 1]
        print('dim no crt', df1.shape)
        print('dim crt:', df2.shape)
        return df1, df2
    else:
        return df

def clinical_data(crt=False):
    """
    Imports and changes the clinical data into desired format

    Parameters
    ----------
    None

    Returns
    ---------
    dataframe
        clinical data in desired format
    """
    df = load_clinical_data()
    df = clean_data(df)
    df = df.drop(index=[43],)
    return remove_nan_row(df, 5, crt)


if __name__ == "__main__":
    fileloc_1 = "Set1_Combined_v2.csv" #set 1 dataset
```

```
fileloc_2 = "Set2_Combined_v2.csv" #set 2 dataset
clinical = clinical_data()
mri_1 = MRI_data(fileloc_1)
mri_2 = MRI_data(fileloc_2)

crt_n_1, crt_y_1 = MRI_data(fileloc_1, crt=True)
crt_n_2, crt_y_2 = MRI_data(fileloc_2, crt=True)
clin, clin_crt = clinical_data(crt=True)
```

B.3 Feature reduction

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 22 23:39:30 2021

@author: Marthe Søvdsnes
"""

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from RENT import RENT
import warnings
warnings.filterwarnings("ignore")


def rent(x_train, x_test, y_train):
    """
    Feature selection using RENT

    Parameters
    ----------
    x_train: df
        the training data
    x_test: df
        The test data
    y_train: series
        respons variables to x_train

    Returns
    ---------
    dataframe
        train data with the only the selecte features
    dataframe
        test data with the only the selected features

    """
    C_param = [0.1, 0.5, 1]
    l1_ratio = [0, 0.1, 0.25, 0.5, 0.75, 0.9, 1]

    y_train = y_train.to_numpy()
    model = RENT.RENT_Classification(data=x_train,
                                     target=y_train,
                                     feat_names=x_train.columns,
                                     C=C_param,
                                     l1_ratios=l1_ratio,
                                     poly='OFF',
                                     testsize_range=(0.25,0.25),
                                     scoring='mcc',
                                     K=100,
                                     random_state=0,
                                     verbose=0)

    model.train()
    selected_features = model.selectFeatures(tau_1_cutoff=0.20,
                                             tau_2_cutoff=0.20,
                                             tau_3_cutoff=0.8)

    x_train = x_train[x_train.columns[selected_features]]
    x_test = x_test[x_test.columns[selected_features]]
    return x_train, x_test
```

```
def pca(x_train, x_test, n):
    """
    Feature extraction using PCA

    Parameters
    ----------
    x_train: df
        the training data
    x_test: df
        The test data
    n: int
        the number of principal conponents

    Returns
    ---------
    dataframe
        train data with the new pca components as columns
    dataframe
          test data with the new pca components as columns

    """


    scaler = StandardScaler()
    x_train = scaler.fit_transform(x_train)
    x_test = scaler.transform(x_test)

    pca = PCA(n_components=n)
    x_train = pca.fit_transform(x_train)
    x_test = pca.transform(x_test)
    return x_train, x_test
```

B.4 Workflow

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 26 20:29:01 2021

@author: Marthe Søvdsnes
"""

import pandas as pd
from mr_data import MRI_data, clinical_data
from algorithms import KNN, RF, sv, LR
from feature_extraction import pca, rent
from sklearn.model_selection import StratifiedKFold
import numpy as np

def run_models(x_train, x_train_pca, x_train_rent, y_train):
    """
    Runs the models

    Parameters
    ----------
    x_train: df
        the training data
    x_train_pca: df
        the training data treated by PCA
    x_train_rent:
        the training data treated by RENT
    y_train: series
        respons variables

    Returns
    ---------
        results from each model as df and best model as estimator
    """

    knn = KNN(x_train, y_train)
    knn_pca = KNN(x_train_pca, y_train)
    knn_rent = KNN(x_train_rent, y_train)
    knn_results = results(knn, knn_pca, knn_rent)

    lr = LR(x_train, y_train)
    lr_pca = LR(x_train_pca, y_train)
    lr_rent = LR(x_train_rent, y_train)
    lr_results = results(lr, lr_pca, lr_rent)

    rf = RF(x_train, y_train)
    rf_pca = RF(x_train_pca, y_train)
    rf_rent = RF(x_train_rent, y_train)
    rf_results = results(rf, rf_pca, rf_rent)

    svc = sv(x_train, y_train)
    svc_pca = sv(x_train_pca, y_train)
    svc_rent = sv(x_train_rent, y_train)
    svc_results = results(svc, svc_pca, svc_rent)

    best = best_model(knn_results, lr_results, rf_results, svc_results)
    return knn_results, lr_results, rf_results, svc_results, best

def pca_rent(x_train, x_test, y_train):
    """
    Creates the RENT and PCA training and test sets
    Parameters
    ----------
    x_train: df
        the training data
    x_test: df
        the test data
    y_train: series
        respons variables

    Returns
    ---------
        training and test sets treated by RENT and PCA as four seperate dataframes
    """

    x_train_pca, x_test_pca = pca(x_train, x_test, 10)
    x_train_rent, x_test_rent = rent(x_train, x_test, y_train)
    return x_train_pca, x_test_pca, x_train_rent, x_test_rent

def best_model(knn, lr, rf, svc):
    """
    Finds the best model

    Parameters
    ----------
        knn: df
            results from all the knn models
        lr: df
            results from all the logistic regression models
        rf: df
            results from all the random forest models
        svc: df
            results from all the SVC models

    Returns
    ---------
        DataFrane
            the model with the higest mean test score
    """

    results = knn.transpose().append([lr.transpose(),
                                      rf.transpose(), svc.transpose()])

    best_score = max(results['Mean Test Score'])
    best = results.loc[results['Mean Test Score']==best_score]
    return best


def results(model, model_pca, model_rent):
    """
    makes the result dataframe for the models

    Parameters
    ----------
        model: estimator
            model trained on training data
        model_pca: estimator
            model trained on training data treated by PCA
        model_rent: estimator
            model trained on training data treated by RENT
    Returns
    ---------
```

```
        dataframe
    """

    results = pd.DataFrame()

    model_all = [model.cv_results_['params'][model.best_index_],
                 model.cv_results_['mean_test_score'][model.best_index_],
                 model.cv_results_['mean_train_score'][model.best_index_],
                 model.cv_results_['std_test_score'][model.best_index_],
                 model.cv_results_['std_train_score'][model.best_index_],
                 model]

    pca = [model_pca.cv_results_['params'][model_pca.best_index_],
           model_pca.cv_results_['mean_test_score'][model_pca.best_index_],
           model_pca.cv_results_['mean_train_score'][model_pca.best_index_],
           model_pca.cv_results_['std_test_score'][model_pca.best_index_],
           model_pca.cv_results_['std_train_score'][model_pca.best_index_],
           model_pca]

    rent = [model_rent.cv_results_['params'][model_rent.best_index_],
            model_rent.cv_results_['mean_test_score'][model_rent.best_index_],
            model_rent.cv_results_['mean_train_score'][model_rent.best_index_],
            model_rent.cv_results_['std_test_score'][model_rent.best_index_],
            model_rent.cv_results_['std_train_score'][model_rent.best_index_],
            model_rent]

    results['All'] = model_all
    results['PCA'] = pca
    results['RENT'] = rent
    results['index'] = ['Params', 'Mean Test Score', 'Mean Train Score',
                        'Std Test Score', 'Std Train, Score', 'Model']
    results = results.set_index('index')
    return results

def k_fold(X, y):
    """
    splits the data up in 4 test folds

    Parameters
    ----------
        X: df
            data from the clinical data set
        y: series
            corresponding respons variables

    Returns
    ---------
        test: list
            list containing the indexes for the test data for each fold
    """

    test = []
    X_index = X.reset_index()
    skf = StratifiedKFold(n_splits=4, random_state=0)
    skf.get_n_splits(X_index, y)
    for train_index, test_index in skf.split(X_index, y):
        temp_test = X_index.iloc[test_index]['index']
        test.append(temp_test)
    return test

def workflow(test_index, X, y):
    """
    splits the full dataset into train and test data, performs RENT and PCA,
    trains the models and find the model with the higest mean test score

    Parameters
    ----------
        test_index: list
            indexes for test data
        X: df
            full dataset
        y: series
            corresponding response variables

    Returns
    ---------
        best: df
            the best model
        x_test: df
            test data
        x_test_pca: df
            test data treated by PCA
        x_test_rent: df
            test data treated by RENT
        y_test: series
            test response variables
    """

    x_train, x_test = X.drop(index=test_index), X.loc[test_index]
    y_train, y_test = y.drop(index=test_index), y.loc[test_index]
    x_train_pca, x_test_pca, x_train_rent, x_test_rent = pca_rent(x_train,
                                                              x_test,
                                                              y_train)
    knn_results, lr_results, rf_results, svc_results, best = run_models(x_train,
                                                                    x_train_pca,
                                                                    x_train_rent,
                                                                    y_train)
    return best, x_test, x_test_pca, x_test_rent, y_test

def voter(pred_1, pred_2, pred_clin):
    """
    makes the majority vote predictions

    Parameters
    ----------
        pred_1: list
            predictions from the best model form set 1
        pred_2: list
            predictions from the best model from set 2
        pred_clin: list
            predictions from the best model from the clinical data

    Returns
    ---------
        vote_pred: list
            predictions from the majority vote classifer
    """

    voter_pred = []
    for i in range(len(pred_1)):
        pred_sum = pred_1[i] + pred_2[i] + pred_clin[i]
        if pred_sum < 2:
                voter_pred.append(0)
            else:
                voter_pred.append(1)
        return voter_pred


if __name__ == "__main__":
    fileloc_1 = "Set1_Combined_v2.csv" #set 1 data
    fileloc_2 = "Set2_Combined_v2.csv" #set 2 data
    clinical = clinical_data()
    mri_1 = MRI_data(fileloc_1, crt=False)
    mri_2 = MRI_data(fileloc_2, crt=False)

    clin = clinical_data(crt=False)
    X = clin.iloc[:, :-2] #all columns exept PFS-event and OS-event
    y = clin['OS-event']
    x_mri_1 = mri_1.iloc[:, :-4]
    y_mri_1 = mri_1['OS-event']
    x_mri_2 = mri_2.iloc[:, :-4]
    y_mri_2 = mri_2['OS-event']

    test_index = k_fold(X, y)

    #change the index in test_index to change fold
    best_1, x_test_1, x_test_pca_1, x_test_rent_1, y_test_1 = workflow(test_index[1], x_mri_1,
    best_2, x_test_2, x_test_pca_2, x_test_rent_2, y_test_2 = workflow(test_index[1], x_mri_2,
    best_clin, x_test_clin, x_test_pca_clin, x_test_rent_clin, y_test_clin = workflow(test_inde

    #change the input test dataset based on which dataset the best model uses
    pred_1 = best_1['Model'][0].predict(x_test_rent_1)
    pred_2 = best_2['Model'][0].predict(x_test_rent_2)
    pred_clin = best_clin['Model'][0].predict(x_test_pca_clin)

    vote_pred = voter(pred_1, pred_2, pred_clin)
```