FYS-STK4155 - Project 3

Jens Junker Pedersen, Fredrik Junker Pedersen

December 19, 2022

Contents

Abstract

The goal in the field of radiomics is to establish links between diagnosis, prognosis and treatment response, from correlations in quantitative medical image features. Image modalities such as Positron Emission Tomography (PET) suffers from high feature variability inter scanners and due to different image reconstruction parameters. Knowledge about how different parameters affects quantitative feature values is needed.

We will try to identify the features that are best able the separate different PET reconstruction method in a systematic way with the use of Support Vector Machine (SVM).

Feature selection is an important step in the machine learning process because it helps to reduce the complexity of the model, improve the accuracy of the model, reduce overfitting, reduce the training time of the model, increase explain ability of the model. Feature selection helps to identify the most important features that contribute to the prediction and discard the irrelevant or redundant features.

In the first part we will use random forest and correlation for feature selection to identify the features with best class separability between reconstruction.

Then we will apply feed forward selection with Suport Vector Machine (SVM) to see if we are able to correctly predict the class targets. And also validate if or prior features selection agrees with the results obtained with SVM.

The features performing best in SVM agreed well with the features showing best sperability in the random forest and correlation analysis for two of the classification combinations. The feature combination of `histogram min` and `glcm homogenity` produced the best class separation between reconstruction methods. With the best overall accuracy of 0.817 for classification of the `PT_PET_EARLAC` versus `PT_PET_WB_Q_CLEAR` reconstruction method, indicating most dissimilarity. In a raiomics setting it's important to be aware that there will be significant variation in those feature values with respect to different reconstruction methods. Our methodical approach for quantifying dissimilarity between reconstruction methods was not suited for the problem, and should not be investigated any further.

1 Introduction

Feature selection is an important step in the machine learning process because it helps reduce the complexity of the model and helps improve its accuracy. By selecting the most relevant features, the model can better focus on the important data points and ignore the noise. This can increase the accuracy of the model and help reduce overfitting. Additionally, feature selection helps reduce the time and resources required to train the model, which can be especially helpful when dealing with large datasets. Selecting good features will make the model more interpretable by removing redundant or irrelevant features.

Radiomics is a field of medical imaging that uses advanced image processing algorithms to extract quantitative features from medical images such as CT, MRI, and PET scans. These features are then used to develop predictive models to better understand a patient's disease and to predict outcomes. Radiomics can be used to better understand the biology of cancer, to develop personalized treatments, and to monitor the effectiveness of treatments.

There are some serious challenges that needs to be solved in order for Radiomics to be clinically relevant. A Radiomics feature should reflect the underlying pathology and not vary due to changes in scanner type or scanner specific parameters. Different scanner types, reconstruction and image processing parameters affect feature values differently. These parameters need to be controlled in order for the field to have any clinical relevance.

By harmonizing the calculation and representation of radiomic features, it is possible to more accurately compare and analyze the features across different studies or datasets, enabling more

reliable and reproducible results in radiomics research.

In this report we will explore a lymphoma image dataset acquired with a PET scanner. A total of n = 16 malignant tumors was identified in the image data Each image is reconstructed with different reconstruction protocols (our class labels). A total of 15 radiomics features is extracted from the tumours (Volume of Interest) We will explore simple machine learning algorithms to explore if it's possible to discriminate between the different reconstruction methods.

If a clear pattern between reconstruction algorithms exists, it may be possible to transform features values between different reconstruction methods. And therefore control for the specific reconstruction methods. In such a way that it may be possible to define absolute standards for the interpretation of a specific feature value and it's significance in terms of pathology. We will not try to harmonize the data, but rather explore if harmonization models it's worth exploring.

In the first part of the report we will implement simple feature selection techniques to identify features that are able to discriminate between the different PET reconstruction methods. We will use random forest to identify the features with best discriminative power. Then we will use person correlation to identify highly correlated redundant features. In the last part we will use feed forward feature selection in conjugation with Support Vector Machine (SVM). We will then evaluate if our prior analysis is in agreement with the results obtained with SVM. If our models and selected features is able to accurately discriminate between the different classes. Then, it may be possible find models that are able to map feature values between series, and define absolute standards for the quantitative meaning of a feature value with respect to pathology.

The first part of the method section (2.1) explains how the data is organized and how to reproduce the data with python code. Subsection 2.2 and 2.3 contains the necessary theory on how to implement a random forest. The last part of subsection 2.3 explains how to reproduce our results obtained with random forest feature selection. And also comparison of our own implementation vs the sci-kit learn python library. The code used to produce our results with the Random Forest algorithm can be found in the Github repository, https://github.com/jensjpedersen/Projects_FYS-STK4155/tree/main/Project3

In subsection 2.5, first the theory of Support Vector Machines is layed out and explained in subsubsection 2.5.1. Then a code example is added to show how the SVM optimization problem might be solved and an explanation of how we will utilize the SVM is given in subsubsection 2.5.2. The code used to produce our results with SVM can be found in the Github repository, https://github.com/fredrikjp/FYS-STK4155/tree/master/Project3.

The first part of our Discussion we present our results and discussion from feature selection with Random Forest and Correlation analysis. Then the results and discussion from our Support Vector Machine is presented for different series classifications.

Finally we summarize and conclude the project in the Conclusion section.

2 Methods

2.1 Dataset

Table 1: List of all features used in analysis. The exact feature definitions can be found in the appendix A

Feature number	Feature name
0	histogram max
1	histogram mean
2	histogram min
3	histogram peak
4	histogram std
5	shape area_density
6	shape convex_hull_area
7	shape surface
8	shape volume
9	shape volume_density
10	glcm cluster
11	glcm contrast
12	glcm entropy
13	glcm homogeneity
14	glcm joint_maximum

Our dataset contains 15 Radiomics features extracted on 16 different lymphom cancers. All the feature values can be found in the Github repository (https://github.com/jensjpedersen/Projects_FYS-STK4155/tree/main/Project3) with path `./Data/big_test.csv`. The name column corresponds to the different tumors (Volume of Interest's), the series column contains the names of a total of six unique PET reconstruction methods. The analysis done in this report uses only tree of the six reconstruction methods, with names PT_PET_EARL2, PT_PET_EARLAC and PT_PET_WB_Q_CLEAR. They will be our target class labels in or supervised classification methods. The respective feature names can be found in the feature_name column with it's respective values in the value column.

To read the data, the following python code can be used.

```
import read_csv
input_path = '../../Data/big_test.csv'
r = read_csv.ReadCSV(input_path)
r._remove_series([2, 3, 5])
X, y = r.get_df()
```

In the above code X is a pandas frame with size $n = 3 \cdot 16$ rows (sample size) and $p = 15$ columns (features). Our target class labels (reconstruction methods) is encoded as a on hot vector, where the first, second and third correspond to our reconstruction methods PT_PET_EARL2 PT_PET_EARLAC PT_PET_WB_Q_Clear respectively. In the cases where integer class labels is used to reference the different target the values 0, 1 and 2 are use in the respective order as above.

The features are listed in table 1. For exact feature definitions see A. All the features is aggregated in a Volume Of Interest (VOI). Each VOI has a threshold of $0.41 \cdot max$. That is, the minimum

value in the VOI is larger or equal to 0.41% of the maximum value voxel (3D pixel) intensity in the tumor. Defining the boundary as a minimum value is meaningful since the voxel intensity value in a PET scanner is positively correlated with cell proliferation. All the images is acquired with a PET Scanner where different image reconstruction techniques was applied to each image. In our analysis we will study 3 different reconstruction methods. This will be our classification targets. Thus, we are interested to see if it is possibly to identify features that gives good class separation with respect to reconstruction method. This will be a challenging since our data only contains 16 samples per reconstruction. Hence, a total of $16 \cdot 3$ data points. This will be an exercise in selecting a few good features to prevent overfitting applied to simple supervised classification models.

2.2 Decision Tree Classifier

A classification decision tree is a predictive model that predicts the target class of a feature based on a set of binary rules. The decision tree consists of a series of nodes, branches, and leaves. Each node represents a decision point, each branch represents a possible outcome, and each leaf represents a final consequence or result.

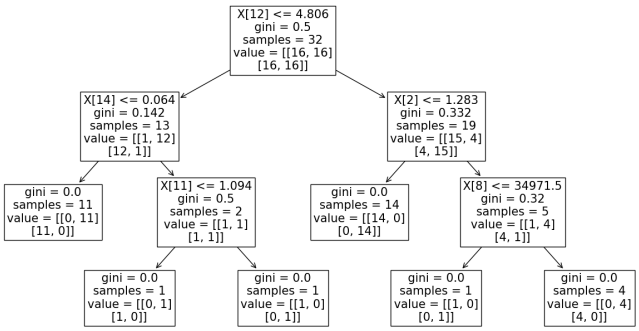

Figure 1: Decision tree

A typical structure of a decision is shown in figure 1. Each rectangle represents a node in the tree. The build of a decision consists of finding the feature and feature value (threshold) at each node that maximizes the purity of each child node resulting from the split. At each node the data is split into two child nodes based on the threshold value. Values lower than the threshold is passed to the left child node and values higher is passed to the right child node. Thus, the tree consists of a set of rules on where to place a specific sample. The algorithm is recursive and starts at the root node (the upper rectangle) on continues until a Leaf node is reached. That is, one of the squares in the bottom of the figures.

2.2.1 The CART algorithm

We implemented the CART algorithm to find the optimal splits in the data.

The CART algorithm splits the data set in two subsets using a single feature k and a threshold t_k [2]. The purity of the split is then evaluated with the cost function:

$$C(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}},$$

where G is a measure of impurity (see section 2.2.2), m_{left} and m_{right} is the number of samples in the left and right subsets resulting from the split. The total number of samples is $m = m_{left} + m_{right}$

The algorithm iterates through all p features and n samples. For each feature k and threshod t_k, the cost score is evaluated on the two subsets. The threshold and features that produced the best score is stored in a node. This process is repeated on the data subsets in a recursive way until a specific criterion is met. This produces a leaf node in the tree.

In our implementation the recursion stops if it is not possible to split the subset further. That is, the number of samples in the subset is equal to one. The recursion will also stop if our subset is 100% pure. That is, the subset only consist of one class label.

2.2.2 Gini factor

The Gini factor is used to measure the purity of the data subset and is calculated as:

$$G = 1 - \sum_{i=1}^{c} P_i^2$$

where P_i is the probability that a sample belongs to class i.

For a maximum impurity, where $P_1 = P_2 = 0.5$ the Gini factor is 0.5. A smaller value indicates a more pure data subset. For a perfectly pure class the Gini factor is 1. That is all the samples belongs to a class k ($P_k = 1$).

Our class target labels was stored in one hot vector of size $n \times c$, where n is the number of samples in the data subset and c is the number of classes. The probability of each class was calculated with this python snippet:

```
import numpy as np
P = np.sum(y, axis = 0)/y.shape[0]
```

, where `y` is the one hot vector of our class targets.

2.2.3 Feature importance

Our random forest implementation was used to identify most discriminative features and not used for its predictive power. Again, we will use the Gini factor to calculate the feature importance.

We want to quantify the impurity decrease as results of a feature split. Therefore, we are interested in the change in impurity with respect to a node and its child nodes. The impurity decrease also needs to take into account the number of samples. A feature is more important if it is able to separate more samples.

We will use the same formula as the sci-kit learn python library to estimate the feature importance (see [1]):

$$\frac{m}{M} \cdot (G - \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}),$$

where G, G_{left} and G_{right} as the Gini factor for the parent, left child and right child node respectively. M is the total number of samples in our input data. The other quantities are as defined in equation 2.2.1.

All our decision trees has a variable called `feature_importances_`, which contains the impurity decrease obtained from the best split of a node (except for leaf nodes).

When building the tree all features is considered each time a node is to be split. If the same feature is used multiple times to split nodes in to a tree. Then, only the split that produces the highest impurity decrease is stored in the `feature_importances_` variable. Hence, used to evaluate the feature importance.

Our decision tree algorithm can be used directly to identify good features. However, there are some problems. The algorithm usually produces very different results dependent on small variations in the data. That is, classification decision trees suffers from high variance. It may also neglect highly correlated features that produces a slightly poorer separation. To circumvent these problems we implemented the random forest algorithm.

2.3 Random forest

Random forest is a type of ensemble learning method, which uses multiple decision trees to make predictions and combines them to create a more accurate and stable prediction.

In our first attempt to identify good features (features with good class separation), we will use a random forest classifier. It can be used to identify the features and feature value that splits the data into the purest classes (see section 2.2.3) The algorithm works by constructing multiple decision trees, each of which is trained on a subset of the data.

It uses a technique called bagging, which involves randomly selecting a subset of data from the training set and building a decision tree from each subset.

It also uses feature resampling to reduce the overfitting of the model. We will implement feature resampling to better evaluate the performance of all the features. Feature resampling is done within each node of the descion tree. Hence, the best split at each node is evaluated on different feature subsets. The pseudocode for building a random forest is shown in Algorithm 1. The algorithm is reused from [2].

Algorithm 1 Growing a Random Forest

Input: training data $\boldsymbol{X}$, number of trees B
Initialize forest $\boldsymbol{T} \leftarrow$
for $b \leftarrow 1$ **to** B **do**
    Draw a bootstrap sample from $\boldsymbol{X}$
    Initialize tree T_b
    while node size $<$ maximum **do**
        Select $m \leq p$ random variables from p predictors/features
        Select best split point among m variables using CART algorithm
        Create new node and split into daughter nodes
    end while
    Add tree T_b to forest $\boldsymbol{T}$
end for
Output: ensemble of trees $\boldsymbol{T}$

We did compare our own implementation of the random forest to the python scikit learn function RandomForestClassifier. For sklearn and our own implementation we used the following parameters, `n_estimators=100000`, `max_features='sqrt'`. Where n_estimators is the number of decision trees in the forest. A total of $\sqrt{p}$ random features was considered in each split, where p=15, is the total number of features. Then the models was fitted to the full dataset. Our fitted classifier will then contain a list of all the fitted decision trees. The feature_importances_ variable obtained from all the decision trees was used to find the overall best features.

The results from our own implementation and sci-learn was in perfect agreement with respect to ranking feature importance. However, the impurity decrease obtained with sklearn was approximately two times larger for all features compared with the results obtained with our own implementation. Our own implementation was also more tan 100 times slower compared to sklearn. An increase in runtime from minutts to hours. This defeats the purpose of our feature selection algorithm. Therefore, the results obtained with the random forest in table 2 is obtained with sklearn's implementation. The following python code can easily be used to reproduce our results:

```
import numpy as np
from sklearn.ensemble import RandomForestClassifier
np.random.seed(0)
RandomForestClassifier(n_estimators=10000, max_features='sqrt')
clf.fit(X, y)
importances = np.zeros((clf.n_estimators, X.shape[1]))
for i, tree_clf in enumerate(clf.estimators_):
    importances[i, :] = tree_clf.feature_importances_
importances = np.mean(importances, axis = 0)
```

The mean importance decrease of all the features was stored in the importances variable. The results obtained with our own implementation is listed in table 5 in the appendix.

2.4 Pearson correlation

Pearson correlation is a statistical measure of the linear relationship between two variables. It is used to assess the strength and direction of the relationship between two continuous variables. The Pearson correlation coefficient ranges from -1 to 1, with values closer to 1 indicating a strong positive relationship, values closer to -1 indicating a strong negative relationship, and values closer to 0 indicating a weak or no relationship.

To evaluate the correlation between features we used the `pandas.DataFrame.corr` functionality from the Pandas python package. We only considered highly positively correlated features when looking for redundant features.

2.5 SVM

2.5.1 SVM theory

Support Vector Machine (SVM) is a machine learning method used for classification and regression problems. We will use SVM for binary classification where the algorithm's aim is to create a $p-1$ dimensional affine subspace of the p dimensional feature space which distinguishes all the classifications optimally. This subspace is called a hyperplane and the optimal hyperplane maximizes the margin which is the distance from the hyperplane to the nearest data point in

feature space. The optimal hyperplane will therefore be the same distance from the nearest point(s) of each classification it divides. These points are called the support vectors.

The hyperplane is defined by a p dimensional weight vector $\boldsymbol{w}$ and a bias term b [3]

$$\boldsymbol{x^T w} + b = 0, \tag{1}$$

where $\boldsymbol{x^T} = [x_1, x_2, ..., x_p]$ is the transpose of a point in feature space. Separating our classifications by $y_i = 1$ or $y_i = -1$, we can classify a data point x_i with our hyperplane as

$$y_i = sign(\boldsymbol{w^T x_i} + b).$$

Our margin defined by our hyperplane and training points x_i with classification y_i becomes the largest M such that

$$\frac{1}{||\boldsymbol{w}||} y_i(\boldsymbol{w^T x_i} + b) \geq M, \forall i = 1, 2, ..., p. \tag{2}$$

If we scale this equation such that $||\boldsymbol{w}|| = 1/M$, finding the maximal margin will be equivalent to minimizing

$$||\boldsymbol{w}||$$

subject to the constraints

$$y_i(\boldsymbol{w^T x_i} + b) \geq 1, \forall i.$$

The optimization of the hyperplane can be solved using Lagrange multipliers. For each constraint formulated as $\phi_k(\boldsymbol{x}) = 0$, we add a Lagrange multiplier $\lambda_k \geq 0$ such that $\frac{\partial f}{\partial x_i} + \lambda_k \frac{\partial \phi}{\partial x_i} = 0$, where $f(\boldsymbol{x})$ with $x_i \epsilon [\boldsymbol{x}]$ is the function we want to minimize subject to the constraints.

Using the Lagrange multipliers, the optimization problem can be solved by minimizing the following Lagrangian function [3]

$$L(\lambda, b, \boldsymbol{w}) = \frac{1}{2}\boldsymbol{w^T w} - \sum_i \lambda_i [y_i(\boldsymbol{w^T x_i} + b)] - 1, \lambda_i \geq 0. \tag{3}$$

Taking the derivatives with respect to $\boldsymbol{w}$ and b we obtain the constraints

$$\frac{\partial L}{\partial b} = -\sum_i \lambda_i y_i = 0, \quad \frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_i \lambda_i y_i \boldsymbol{x_i} = 0.$$

Inserting these constraints in Equation 3, we are able to get rid of the variables $\boldsymbol{w}$ and b from the Lagrangian

$$L = \sum_i \lambda_i - \frac{1}{2} \sum_{ij}^{n} \lambda_i \lambda_j y_i y_j \boldsymbol{x_i^T x_j}, \tag{4}$$

with constraints $\lambda_i \geq 0$ and $\sum_i \lambda_i y_i = 0$. Additionally, the Karush-Kuhn-Tucker condition has to be satisfied

$$\lambda_i > 0 \implies y_i(\boldsymbol{w^T x_i} + b) = 1, \quad y_i(\boldsymbol{w^T x_i} + b) > 1 \implies \lambda_i = 0,$$

such that $\lambda_i > 0$ if and only if $\boldsymbol{x_i}$ is a support vector.

There is often overlap between classifications in feature space. Then, there won't exist a hyperplane able to separate all the classifications. We will implement two ways to deal with this problem.

The first is to introduce slack variables. The slack variables $\varepsilon_i \geq 0$ allows for some misclassification of the training data. The amount of misclassification is governed by the slack constant C which sets a bound on the sum of the slack variables. Utilizing this method turns our SVM in to a so-called soft classifier as opposed to a hard classifier. The Lagrangian stays the same as Equation 4, but with different constraints [3]:

$$\begin{aligned} \sum_i \lambda_i y_i = 0, \quad 0 \leq \gamma_i \leq C, \\ \lambda_i[y_i(\boldsymbol{w^T x_i} + b) - (1 - \varepsilon_i)] = 0, \\ y_i(\boldsymbol{w^T x_i} + b) - (1 - \varepsilon_i) \geq 0. \end{aligned}$$

The second method is to transform the basis of the feature space to obtain better separation between classifications. This is done with a so-called kernel K which transforms the data in Equation 4 by

$$\boldsymbol{x_i^T x_j} \to \phi(\boldsymbol{x_i})^T \phi(\boldsymbol{x_j}) = K(\boldsymbol{x_i}, \boldsymbol{x_j}),$$

where $\phi(\boldsymbol{x})$ is some transformation of the feature space, often to a much higher dimensional space.

Our Lagrangian in Equation 4 expressed as a convex optimization problem in terms of a matrix equation using a kernel K may be written as [3]

$$\frac{1}{2}\boldsymbol{\lambda^T} \begin{bmatrix} y_1 y_1 K(\boldsymbol{x_1}, \boldsymbol{x_1}) & y_1 y_2 K(\boldsymbol{x_1}, \boldsymbol{x_2}) & \dots & y_1 y_n K(\boldsymbol{x_1}, \boldsymbol{x_n}) \\ y_2 y_1 K(\boldsymbol{x_2}, \boldsymbol{x_1}) & y_2 y_2 K(\boldsymbol{x_2}, \boldsymbol{x_2}) & \dots & y_2 y_n K(\boldsymbol{x_2}, \boldsymbol{x_n}) \\ \vdots & \vdots & \ddots & \vdots \\ y_n y_1 K(\boldsymbol{x_n}, \boldsymbol{x_1}) & y_n y_2 K(\boldsymbol{x_n}, \boldsymbol{x_2}) & \dots & y_n y_n K(\boldsymbol{x_n}, \boldsymbol{x_n}) \end{bmatrix} \boldsymbol{\lambda} - \mathbb{1}\boldsymbol{\lambda}, \tag{5}$$

where $\mathbb{1} = [1, 1, ..., 1]$ is n dimensional. Here $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_n]^T$ and $\boldsymbol{y} = [y_1, y_2, ..., y_n]^T$. The constraints can be written as $\boldsymbol{y^T \lambda} = 0$, and $0 \leq \lambda_i \leq C$ assuming we are using a slack constant C.

There are many popular kernels such as the Gaussian radial basis function which we will use:

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = exp(-\gamma ||\boldsymbol{x_i} - \boldsymbol{x_j}||^2). \tag{6}$$

Here, γ is a constant which needs to be tuned. We know that the feature space transformation ϕ defining the kernel exists based on Mercer's theorem which states that if K is symmetric, continuous and leads to a positive semi-definite matrix P defined in Equation 5, then ϕ exists [3].

2.5.2 SVM method

To minimize Equation 5, we will use `solvers.qp` from the python library `cvxopt`. The solver takes the minimization problem and its constraints as matrix equations:

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \quad & \frac{1}{2}\boldsymbol{\lambda^T P \lambda} + \boldsymbol{q^T \lambda}, \\ \text{subject to} \quad & \boldsymbol{G\lambda} \leq \boldsymbol{h}, \quad \boldsymbol{A\lambda} = \boldsymbol{b}. \end{aligned} \tag{7}$$

The problem in Equation 7 are defined and solved based on Equation 5 in the following code:

```
def solve(self):
        # Solution for minimizing
        # (1/2) * lambda.T @ P @ lambda + q.T @ lambda
        # subject to constraints

        y = self.y_train
        X = self.X_train
        n = self.X_train.shape[0]

        q = -1 * np.ones(n).T
        P = np.zeros((n, n))
        for i in range(n):
            for j in range(n):
                P[i,j] = y[i] * y[j] * self.K(X[i,:], X[j,:])

        # G and h sets constraints on col vec lambda: G@lambda <= h
        G = np.zeros((2*n, n))
        G [:n, :] = np.identity(n) * (- 1)
        G[n:,:] = np.identity(n)
        h = np.zeros(2*n)
        h[n:] = np.ones(n) * self.C
        # A and b sets  constraints on labda: A@lambda = b
        A = np.array(self.y_train, dtype=float).T
        b = np.zeros(1)

        # Solve
        P, q, G, h, A, b = matrix(P), matrix(q), matrix(G), matrix(h), matrix(A),
    matrix(b)
        solvers.options['show_progress'] = False
        sol = solvers.qp(P, q, G, h, A, b)
        self.lmb = np.array(sol["x"])
        self.lmb_non_zero_indecies = np.where(self.lmb > 1e-5)[0]
        self.calc_b()
```

Having obtained our Lagrange multipliers, we are able to calculate our bias term

$$b = \frac{1}{N_s} \sum_{j \epsilon N_s} (y_j - \sum_{i=1}^{n} \lambda_i y_i K(\boldsymbol{x_i}, \boldsymbol{x_j}),$$

where N_s is the number of support vectors with indices j. Then, we are able to predict the classification of a point x_i in feature space by

$$y_i = sign(b + N_s \sum_{j \epsilon N_s} y_j \lambda_j K(\boldsymbol{x_i}, \boldsymbol{x_j})$$

We will explore which features that are most capable of seperating the classifications by using the SVM model first on every feature pair. Then we will explore the accuracy with the addition of all different third features added to the best performing feature pair. As our dataset consists of 15 features, this is a lot of combinations such that insted of tuning the parameters for each feature combination, we will just tune them using all the feature and use the resulting optimal parameters for every combination.

2.6 Cross Validation

We will use k-fold cross validation with multiple cycles to increase the precession of the test accuracy score. The steps of k-fold cross-validation is:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups.
3. For each unique group:
    a. Decide which group to use as a set for test data
    b. Take the remaining groups as a set for training data
    c. Fit a model on the training set and evaluate it on the test set
    d. Retain the evaluation score and discard the model
5. Summarize the model using the sample of model evaluation scores

The advantage of cross-validation is that the data splitting is not done randomly so we don't get any unwanted influence on the model building or prediction evaluation. This is called a k-fold cross-validation structures the data splitting involves dividing the samples k more or less equally sized exhaustive and mutually exclusive subsets. In turn (at each split) one of these subsets plays the role of the test set while the union of the remaining subsets constitutes the training set. Such a splitting warrants a balanced representation of each sample in both training and test set over the splits.

Multiple cycle cross-validation is a variant of cross-validation that involves repeating the process multiple times, each time using a different combination of the data as the training and evaluation sets.

3 Discussion

3.1 Feature selection

Table 2: Ranking of feature importance from best to worst with respect to impurity decrease for three different reconstruction methods on the full dataset obtained with sci-kit learn.

Rank	Feature index	Feature name	Impurity decrease
0	2	histogram min	0.102915
1	12	glcm entropy	0.088981
2	11	glcm contrast	0.083777
3	4	histogram std	0.081181
4	13	glcm homogeneity	0.080463
5	5	shape area_density	0.073690
6	3	histogram peak	0.069483
7	0	histogram max	0.064857
8	14	glcm joint_maximum	0.061548
9	9	shape volume_density	0.055089
10	8	shape volume	0.051088
11	7	shape surface	0.048897
12	10	glcm cluster	0.048494
13	1	histogram mean	0.047758
14	6	shape convex_hull_area	0.041778

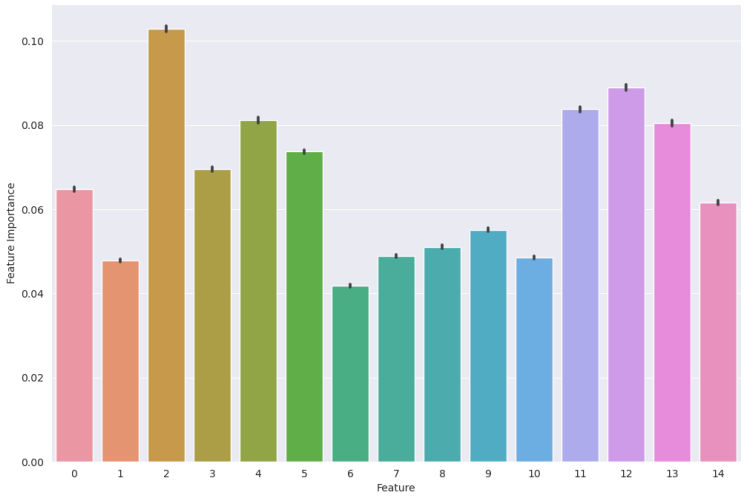

Figure 2: Mean of the feautre_importacnes_ (purity decrease) for each decision tree. The error bar shows the 95% confidence interval. The x-axis represent feature number as defined in table 4.

Figure 2 shows the histogram the feature importance on the y axis and feature index on the x axis. These results was obtained with with sci-kit learn's implementation of the random forest. The 95 % confidence interval is shown on the top of each bar. The histogram min is best able to discriminate between the tree different reconstruction methods, with a impurity decrease of 0.1029&. The numerical feature importance (Impurity decrease) values for all the features is listed in table 2. The glcm entropy features gives the second best impurity decrease with a value of 0.0890. Our next 3 best features are glcm contrast, histogram std and glcm homogeneity which performs similarity with an impurity descrese in the rande 0.0805-0.0838.

In figure 3 the correlation between all the features is plotted. The color bar has a range from -1 (black) to 1 (white), where a score of 1 indicates perfect correlation. The histogram min features is not strongly correlated with any other features, thus we expect this features give a good accuracy on the test data with the SVM model. Our next best features glcm entorpy is highly correlated with histogram std. Thus, only one of the features should be used when training a classification model. Utilizing both features will produce high variance due to redundant information. When training our SVM we therefore expect the combinations of these two features to produce worse accuracy on the test data compared with any other combination of the top 5 features. Glcm contrast has the highest correlation with histogram std, but not to the same extent as the two previous discussed features. GLCM homogeneity has the highest correlation with glcm joint_maximum. However, the GLCM maximum features performance significantly worse with a difference impurity decrease of 0.0189. In our training of the SVM we expect that a combination of hisogram min, glcm entropy. The 4 combinations of features we expect to produce the best predication on the test data is listed in table 3. Also, every combination of two features in each row is expected perform well with SVM.

Table 3: Each row corresponds to a set of feature combinations that is expected to give good accuracy with SVM. Combinations of two of the features in each row shoud also give good accuracy with SVM. The columns is sorted with respect to impurity decrease, where the left most column has the highest impurity decrease

feature 1	feature 2	feature 3
histogram min	gclm entropy	glcm homogeneity
histogram min	gclm entropy	histogram std
histogram min	gclm contrast	glcm homogeneity
histogram min	gclm contrast	histogram std

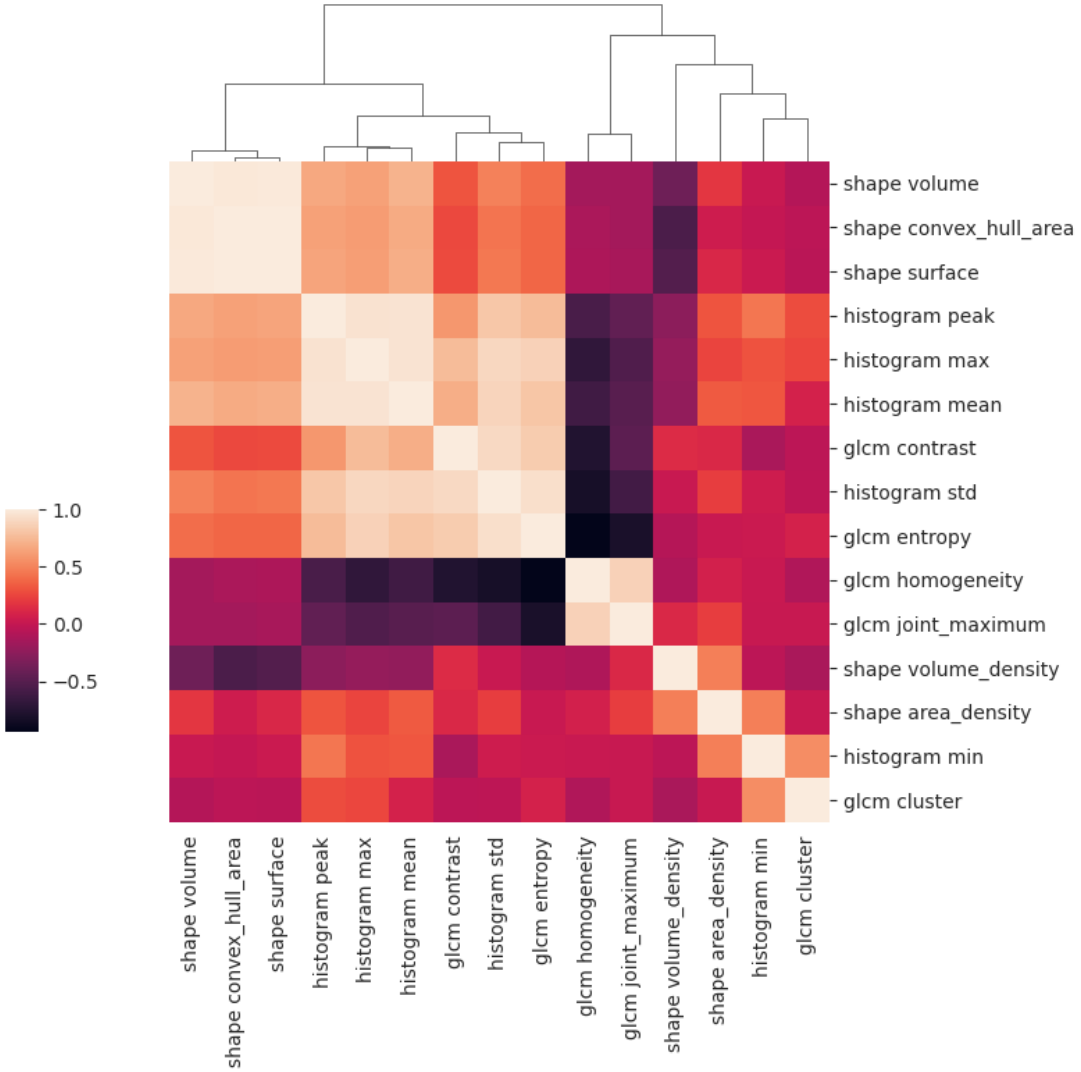

Figure 3: Pearson Correlation between each feature calculated on the full dataset.

3.2 SVM

3.2.1 Classification of series PT-PET-EARLAC vs PT-PET-WB-Q-CLEAR

This subsubsection provides results from the SVM classifying series 1 named `PT_PET_EARLAC` versus series 2 named `PT_PET_WB_Q_CLEAR`.

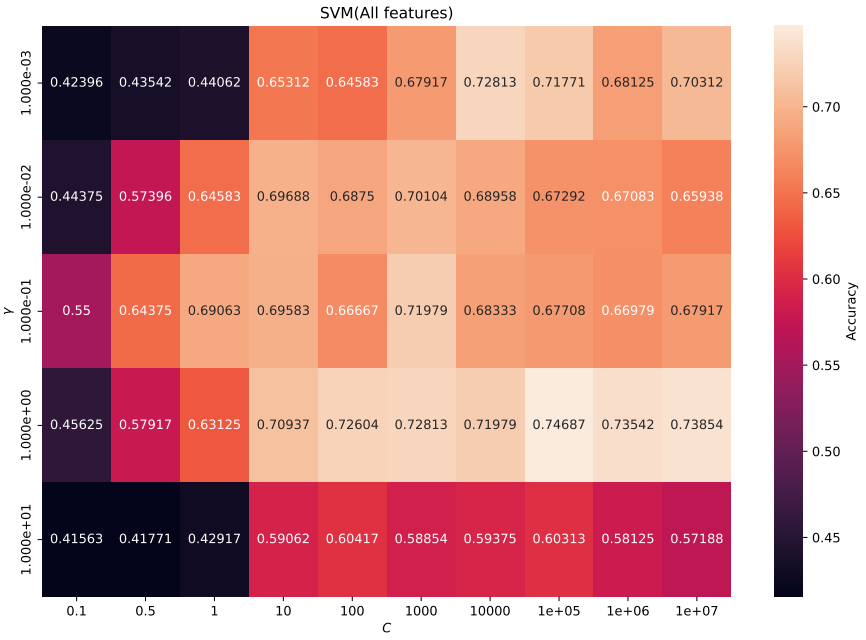

Figure 4: Heatmap of the accuracy obtained with different slack constants C and GRBF kernel factors γ in the SVM model training with all the features. The SVM accuracies are the average of 30 cross-validation cycles training and predicting test data of relative size 0.25. The predictions are wither the data belongs to series 1 or 2.

Figure 4 shows the accuracy of the SVM model's test data predictions for different slack constants C and GRBF kernel factors γ. We observe the best accuracy 0.747 for $C = 10^5$ and $\gamma = 1$. Lower slack constant allows for more misclassification while lower γ results in a wider GRBF kernel which means that the influence of each data point on the hyperplane position will decrease. Both leads towards under fitting on the under-over fit spectrum. Our optimal values for C and γ are quite large suggesting the need for a complex hyperplane to classify our data.

We will utilize these optimal value in our analysis of predictions based on two and three features. They should be tuned for every feature combination, but this would required way to much computation to handle for our computer. The optimal parameter values obtained using all the features will be good general parameters for the following feature combination analysis.

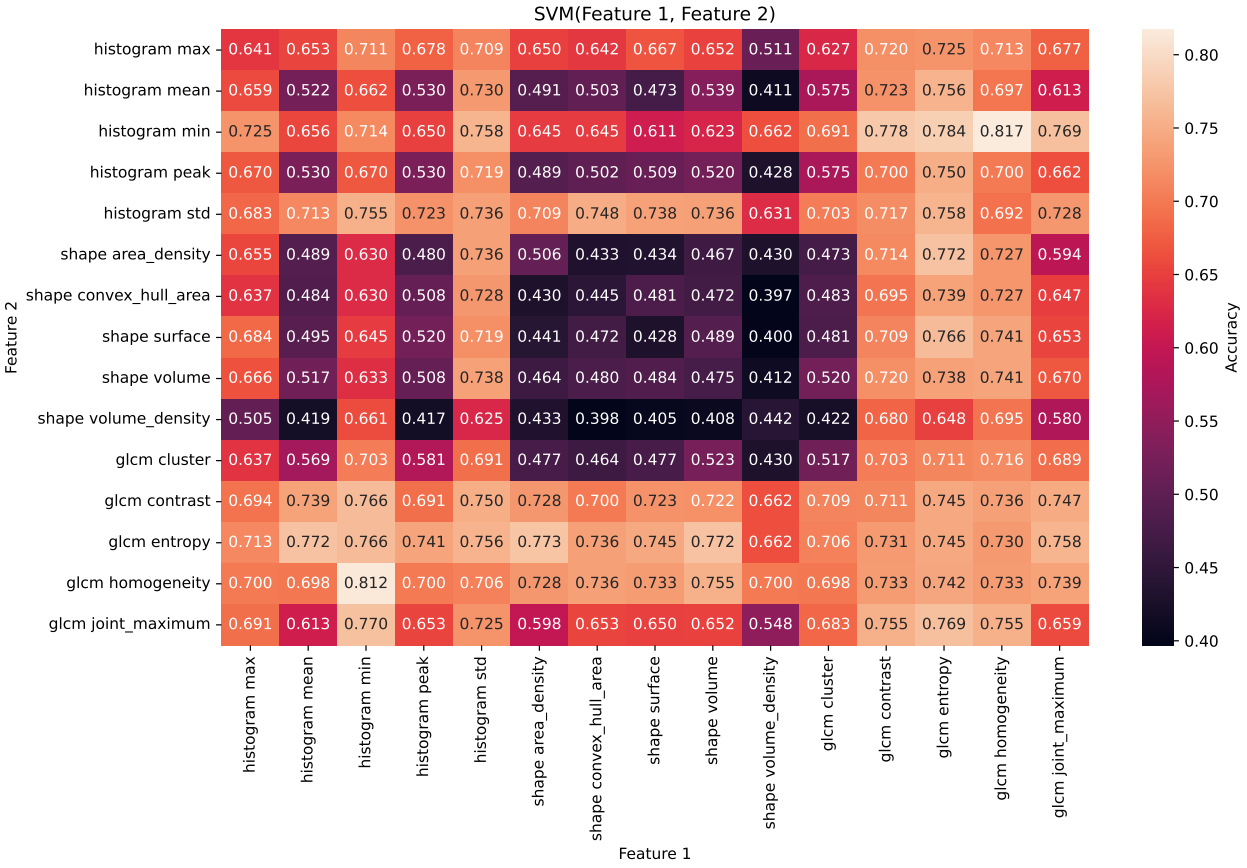

Figure 5: Heatmap of the accuracy obtained by the SVM model training on different feature pairs. The SVM accuracies are the average of 20 cross-validation cycles training and predicting test data of relative size 0.25. The slack constant and GRBF kernel factor are set to $C = 10^5$ and $\gamma = 1$ respectively. The predictions are wither the data belongs to series 1 or 2.

Figure 5 shows a heatmap of the accuracy obtained with different feature pairs utilized by the SVM model for training and predicting. The diagonal from upper left to lower right of the heatmap shows pairs of same features which is equivalent to classification based on that single feature. We observe the best score for a single feature with `glcm entropy` with accuracy 0.745.

The accuracy of each feature pairs have been derived twice, one on either side of the equal feature diagonal, which is nice as this gives some conformation. We observe both scores of the feature pair `histogram min` and `glcm homogeneity` to be the best with accuracy 0.812 and 0.817. This feature pair is one of the combinations expected to give good results in Table 3. The table also shows an expected good result with the feature pair `glcm entropy` and `glcm homogeneity`, although our SVM analysis shows worse accuracy when including `glcm homogeneity` compared to `glcm entropy` alone. Assuming these features have low correlation as shown in Figure 3, one would expect an increase in accuracy as both shows good accuracy on their own. Therefore the poor accuracy might be a result of bad parameters (C and γ) for this combination of feature. The rest of the feature pair combinations in the rows of Table 3 seem to improve the accuracy.

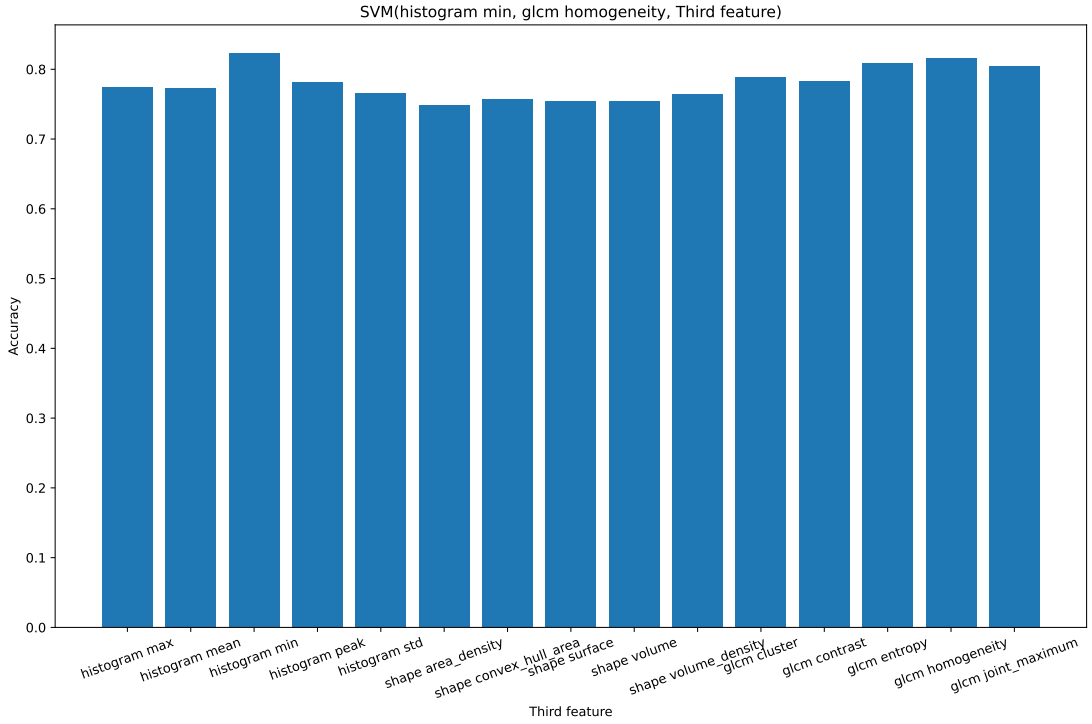

Figure 6: Bar plot of the accuracy obtained adding a third feature in the data used by the SVM model. The accuracies are the average of 50 cross-validation cycles training and predicting test data of relative size 0.25. The slack constant and GRBF kernel factor are set to $C = 10^5$ and $\gamma = 1$ respectively. The predictions are wither the data belongs to series 1 or 2.

Figure 6 shows the accuracy gotten when adding a third feature to the best scoring feature pair `histogram min` and `glcm homogeneity` from Figure 5. We observe best scores for third feature `histogram min` and `glcm homogeneity` which corresponds to no third feature. Thus, the addition of a third feature results in worse accuracy. The optimal feature pair in got better accuracy than what was obtained using all the features in Figure 4 for which the parameters was tuned. This suggest that there will be a point where adding more feature will decrease the prediction power of our model. Our results suggesting this point is for our two features should again be confirmed tuning the parameters C and γ for each third feature.

3.2.2 Classification of series PT-PET-EARL2 vs PT-PET-WB-Q-Clear

This subsubsection provides results from the SVM classifying series 0 named `PT_PET_EARL2` versus series 2 named `PT_PET_WB_Q_CLEAR`.

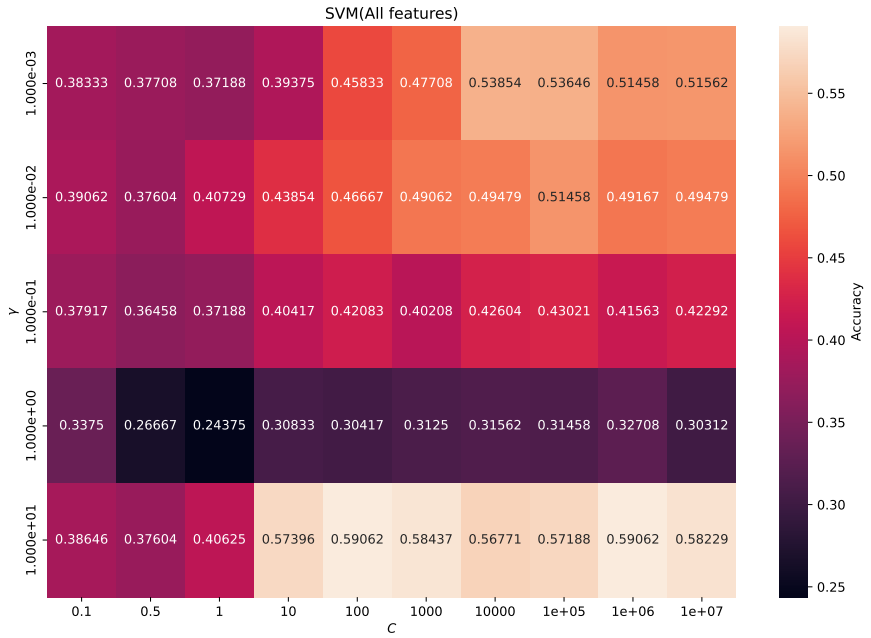

Figure 7: Heatmap of the accuracy obtained with different slack constants C and GRBF kernel factors γ in the SVM model training with all the features. The SVM accuracies are the average of 30 cross-validation cycles training and predicting test data of relative size 0.25. The predictions are wither the data belongs to series 0 or 2.

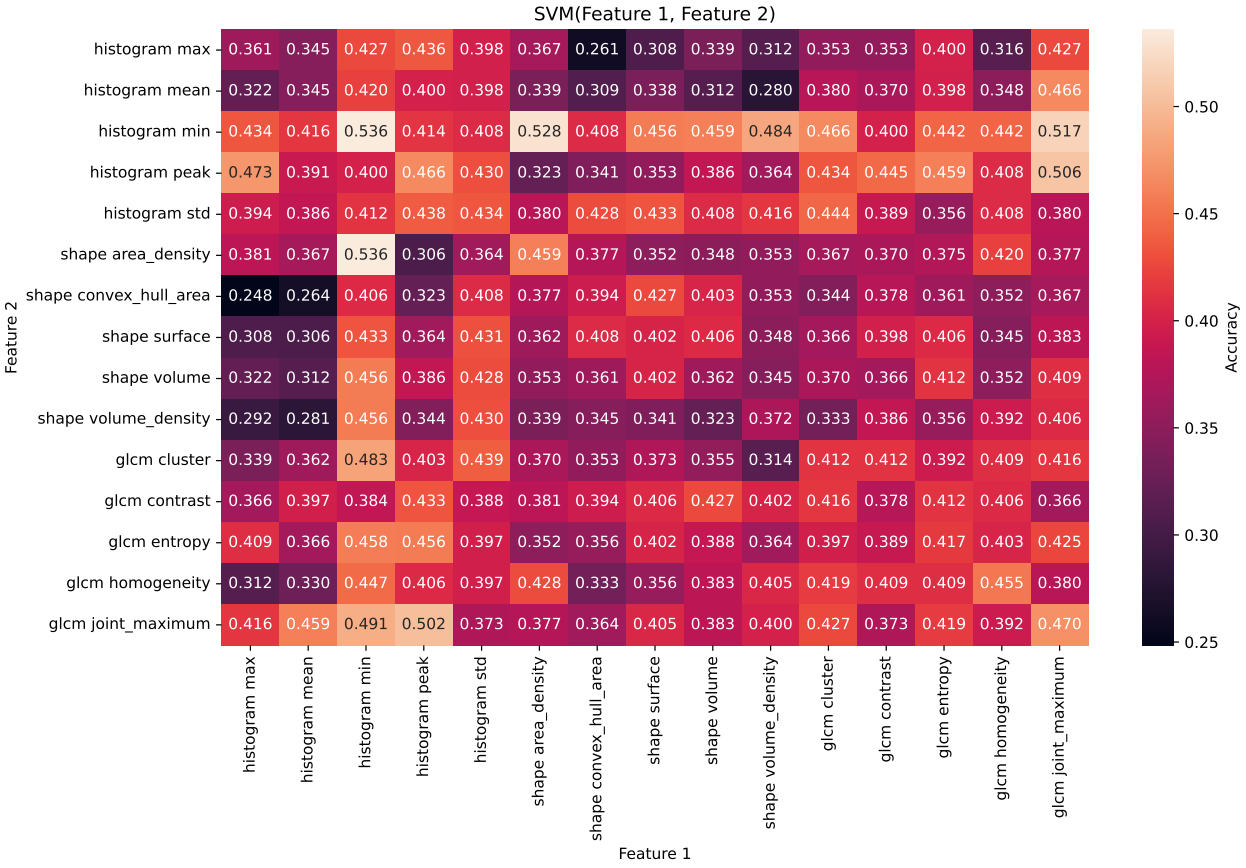

Figure 8: Heatmap of the accuracy obtained by the SVM model training on different feature pairs. The SVM accuracies are the average of 20 cross-validation cycles training and predicting test data of relative size 0.25. The slack constant and GRBF kernel factor are set to $C = 100$ and $\gamma = 10$ respectively. The predictions are wether the data belongs to series 0 or 2.

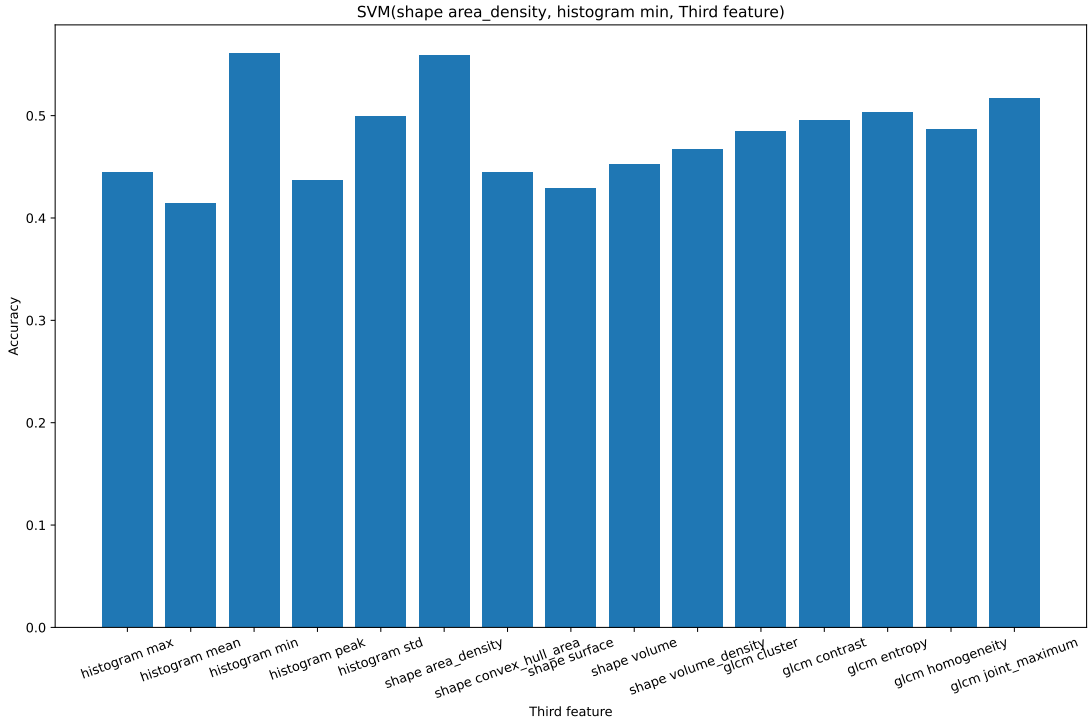

Figure 9: Bar plot of the accuracy obtained adding a third feature in the data used by the SVM model. The accuracies are the average of 50 cross-validation cycles training and predicting test data of relative size 0.25. The slack constant and GRBF kernel factor are set to $C = 100$ and $\gamma = 10$ respectively. The predictions are wether the data belongs to series 0 or 2.

Figure 7-Figure 9 shows the same figures as in subsubsection 3.2.1, but now classifying series 0 versus series 1 instead of series 1 versus series 2.

Figure 7 shows the best accuracy of 0.591 for $C = 100$ and $\gamma = 10$. Utilizing these parameters, Figure 8 shows that the best feature pair accuracy 0.536 was gotten with the feature pair `shape area_density` and `histogram min` where the latter also gave best accuracy 0.536 when only using one feature. From Figure 9, we observe the addition of a third feature results in worse accuracy. The accuracies from single and pair features in Figure 8 are worse than what was obtain with all features.

These results suggests the need for tuning the parameters for each feature combination utilized. This is especially visible in Figure 8 as the worst accuracies are further from accuracy 0.5 than the best accuracies. This is because a useless model would have expected accuracy 0.5 as each classification contains 50% of the data. The trend is also visible in Figure 7 such that the parameters when using all features should be more fine tuned. We may actually use the feature combinations giving the worst accuracies as the most predictive combinations by just classifying opposite to the model classification. Taking this into account we find the feature pair `shape convex_hull_area` and `histogram max` as well as the single feature `histogram mean` to be most predictive with accuracies $1-0.248 = 0.752$ and $1-0.345 = 0.655$ respectively. Also, the best most predictive parameters in Figure 7 are actually $C = 1$ and $\gamma = 1$ leading to a accuracy of $1 - 0.244 = 0.756$.

The features leading to best accuracy in Figure 8 are not expected to perform well from the feature selection analysis in subsection 3.1. The implications of this are discussed in subsubsection 3.2.4.

3.2.3 Classification of series PT-PET-EARL2 vs PT-PET-EARLAC

This subsubsection provides results from the SVM classifying series 0 named `PT_PET_EARL2` versus series 1 named `PT_PET_EARLAC` .

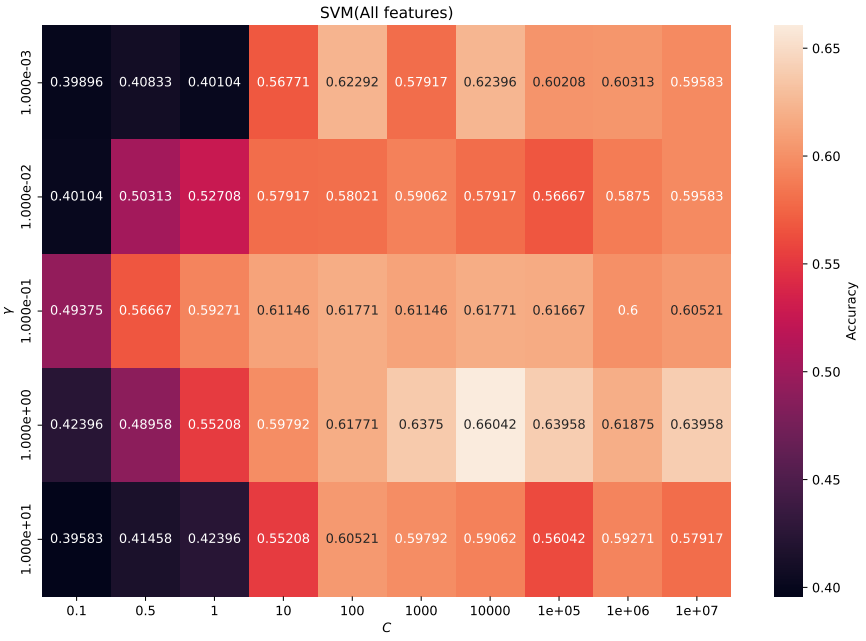

Figure 10: Heatmap of the accuracy obtained with different slack constants C and GRBF kernel factors γ in the SVM model training with all the features. The SVM accuracies are the average of 30 cross-validation cycles training and predicting test data of relative size 0.25. The predictions are whether the data belongs to series 0 or 1.

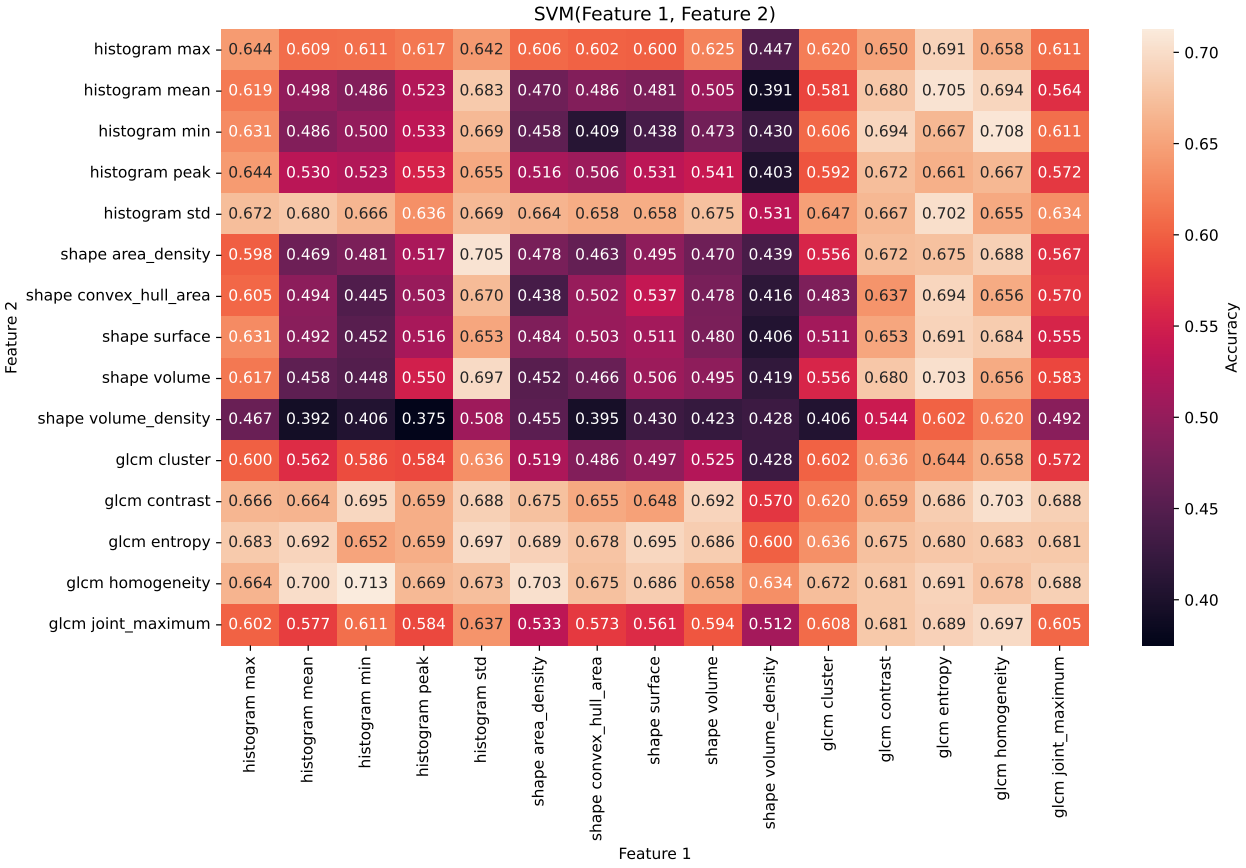

Figure 11: Heatmap of the accuracy obtained by the SVM model training on different feature pairs. The SVM accuracies are the average of 20 cross-validation cycles training and predicting test data of relative size 0.25. The slack constant and GRBF kernel factor are set to $C = 10^4$ and $\gamma = 1$ respectively. The predictions are wither the data belongs to series 0 or 1.

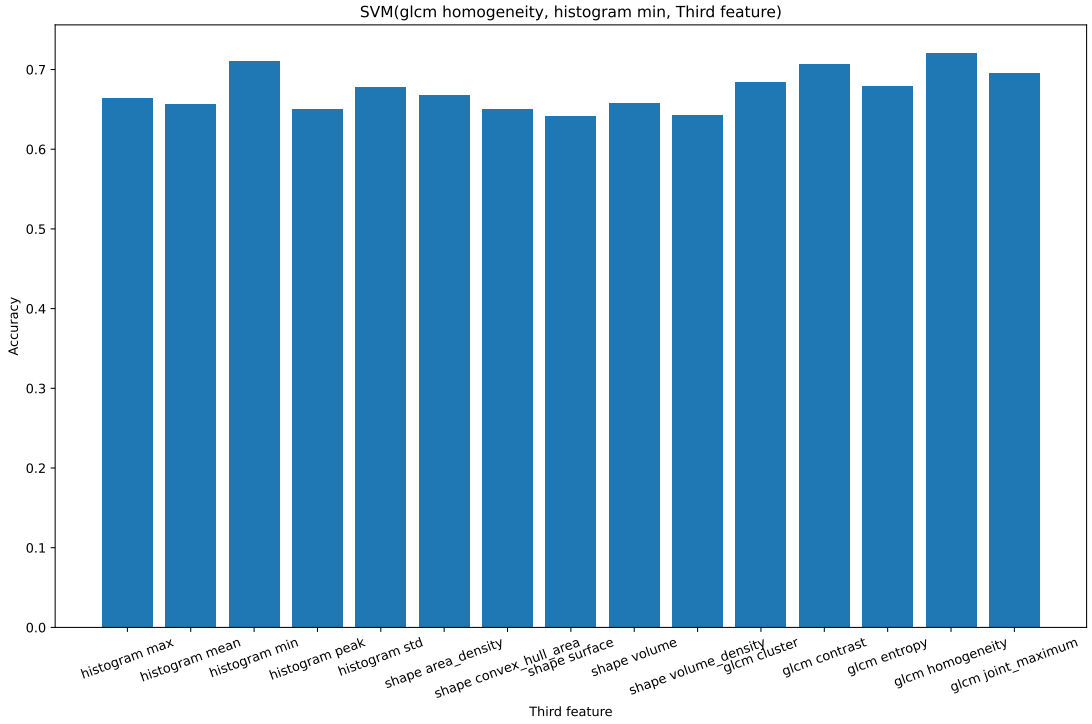

Figure 12: Bar plot of the accuracy obtained adding a third feature in the data used by the SVM model. The accuracies are the average of 50 cross-validation cycles training and predicting test data of relative size 0.25. The slack constant and GRBF kernel factor are set to $C = 10^4$ and $\gamma = 1$ respectively. The predictions are wither the data belongs to series 0 or 1.

Figure 10-Figure 12 shows the same figures as in subsubsection 3.2.1 and subsubsection 3.2.2, but now classifying series 0 versus series 1. From Figure 10 observe the best accuracy 0.660 for the parameters $C = 10^4$ and $\gamma = 1$. Similarly to the analysis classifying series 1 and series 4 in subsubsection 3.2.1, the best feature pair is `Histogram min` and `glcm homogeneity` and the best single feature is `glcm entropy` shown in Figure 11. The addition of a third feature is also similarly resulting in worse accuracy as seen from Figure 12.

3.2.4 Series comparison

The results in subsubsection 3.2.3 classifying series 0 and 1 are very similar to the results in subsubsection 3.2.1 classifying series 1 and 2 although with worse accuracy. They also agree with the analysis done in subsection 3.1. The results from subsubsection 3.2.2 classifying series 0 and 2 are very different, as it shows separation for different features which agrees poorly with the anlysis done in subsection 3.1. This suggest that series 1 is most different and has shifts in some of its features which may be corrected to obtain similar values to series 0 and 2. Series 0 and 2 are more similar, but as seen in subsubsection 3.2.2, these series also have differences in feature values which can be corrected.

4 Conclusion

From our feature selection analysis we identified the 5 best features in terms of impurity decrease. The top features was histogram min, glcm entropy, glcm contrast, histogram std and glcm homogeneity.

We then predicted the test accuracy with SVM. Classification of the target labels PT-PET-EARLAC vs PT-PET-WB-Q-CLEAR and PT-PET-EARL2 vs PT-PET-EARLAC with only one predictor, produced the best accuracy for the features identified in our prior feature selection analysis. Classification of PT-PET-EARLAC vs PT-PET-BB-Q-CLEAR produced an accuracy score in the range 0.711-0.745. Classification of PT-PET-EARL2 vs PT-PET-EARLAC produced an accuracy score in the range 0.644-0.680 for the above five features.

Our feature selection method was effective in selecting the best features for our SVM when only one predictor was considered. This was not the case for classification of the targets PT-PET-EAR2 vs PT-PET-WB-Q-CLEAR, which generally performed poorly on all the features.

The best accuracy obtained with SVM was classification of `PT_PET_EARLAC` versus `PT_PET_WB_Q_CLEAR` using only two features `histogram min` and `glcm homogenity` which provided an accuracy of 0.817.

The optimal accuracy gotten for classification of `PT_PET_EARL2` versus `PT_PET_EARLAC` was 0.713, obtained using the same feature pair. This was one of the feature pair combinations we expected to produce the best results with SVM, based on our prior feature selection analysis (see table 3)

Both classification combinations showed worse accuracy for every third feature added to the optimal pair of features.

The results from the last classification combination `PT_PET_EARL2` versus `PT_PET_WB_Q_CLEAR` was quite different. We observed a more fine tuned parameter selection was needed as we got more predictive power with the lowest accuracies by predicting the opposite of the model. By this method we found `shape convex_hull_area` and `histogram max` to be the feature pair giving the best accuracy 0.752, although the analysis of this classification should be repeated with better tuned parameters.

Comparing the results from different classification combinations, we hypothesized the `PT_PET_EARLAC` series to be most different from the other series.

The `histogram min` and `glcm homogenity` features produced the best class separation between reconstruction methods. When considering such features in a raiomics setting it's important to be aware that there will be significant variation in those feature values with respect to different reconstruction methods From the comparison of our accuracy score we conclude that `PT_PET_EARLAC` and `PT_PET_WB_Q_CLEAR` is the least similar reconstruction methods in terms of feature values. And `PT_PET_EARL2` versus `PT_PET_WB_Q_CLEAR` most similar.

To use supervised learning methods to describe class separability is not a good methodological approach. The purpose of supervised classification method is to correctly identify class labels from an unseen dataset. The problem of exploring class separability and choice of dataset is related to Jens J. Pedersen's master thesis. Thus, the motivation for the unconventional use of M.L. Methods. In a medical setting the labels of the VOI'S and Reconstruction methods is always known. Therefore, simpler methods should be utilized when analysing differences between PET reconstruction methods. We did not manage to find any relevant literature on how the different features compares with respect to the different series. Thus, the validity of the results is uncertain.

We successfully was able to use feature selection techniques to identify the features that produced the best class separation with SVM. Our feature selection methods is relevant in conjunction with SVM and will reduce the computational cost associated with training a model on a large amount of features. Our methodological approach for exploring class separation should not be explored any further.

References

[1] scikit-learn developers. *sklearn tree DescionTreeClassifier*. "https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier". 2022.

[2] Morten H. J. *Week 44: Decision Trees, Ensemble methods and Random Forests*. "https://compphysics.github.io/MachineLearning/doc/pub/week44/html/week44.html". 2022.

[3] Morten H. J. *Week 46: Support Vector Machines and Project 3*. "https://compphysics.github.io/MachineLearning/doc/pub/week46/html/week46.html". 2022.

A Feature Definitions

The definitions of all the features used in this report can be found at the IBSI manual in the following URL; https://ibsi.readthedocs.io/en/latest/03_Image_features.html There are some differences in feature names in this report and the IBSI manual. All the feature names used in this report with the corresponding names used in the IBSI manual is listed in table 4.

Table 4: List of feature names referenced in this report and respective naming used in the IBSI reference manual

Feature index	Feature name report	Feature name IBSI
0	histogram max	Intensity Histogram Maximum
1	histogram mean	Intensity Histogram Mean
2	histogram min	Intensity Histogram Minimum
3	histogram peak	Local intensity Peak
4	histogram std	nan
5	shape area_density	Area density (convex hull)
6	shape convex_hull_area	Morphology, Volume desnity (convex hull)
7	shape surface	Morphology, Surface area (mesh)
8	shape volume	Morphology, Volume (Voxel counting)
9	shape volume_density	Morphology, Volume desnity (convex hull)
10	glcm cluster	Co-occurrence matrix (3D, averaged), Cluster shade
11	glcm contrast	Co-occurrence matrix (3D, averaged), Contrast
12	glcm entropy	Co-occurrence matrix (3D, averaged),
13	glcm homogeneity	Co-occurrence matrix (3D, averaged), Inverse difference moment
14	glcm joint_maximum	Co-occurrence matrix (3D, averaged), Joint maximum

Table 5: Ranking of feature importance from best to worst with respect to impurity descrease for three different reconstruction methods on the full dataset. Thise reults was obtained with our own implementation of the Random Forest algorithm.

Rank	Feature index	Feature name	Impurity decrease
0	2	histogram min	0.058951
1	12	glcm entropy	0.049889
2	11	glcm contrast	0.045033
3	4	histogram std	0.043781
4	13	glcm homogeneity	0.043493
5	5	shape area_density	0.039483
6	3	histogram peak	0.035173
7	0	histogram max	0.032268
8	14	glcm joint_maximum	0.031063
9	9	shape volume_density	0.029351
10	8	shape volume	0.026613
11	7	shape surface	0.025270
12	10	glcm cluster	0.024869
13	1	histogram mean	0.023331
14	6	shape convex_hull_area	0.021340

B Results