TAMPERE UNIVERSITY OF TECHNOLOGY

PINJA KUOSMANEN

DATA MINING ON DEFLECTOMETRIC DATA OF SURFACE DEFECTS

Master of Science Thesis

Examiners: Tapio Elomaa,
Simo Ali-Löytty
Examiners and topic approved
9.8.2017

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Teknis-luonnontieteellinen
Pinja Kuosmanen: Tiedonlouhintaa pintavauriodatalla
Diplomityö, 70 sivua, 5 liitesivua
Elokuu 2017
Pääaine: Matematiikka
Tarkastajat: Tapio Elomaa, Simo Ali-Löytty
Avainsanat: pintavaurioiden havainnointi, maalipinnan automaattinen laaduntarkkailu, koneoppiminen, tiedonlouhinta

Auton maalipinnoilta löytyy usein pieniä virheitä, jotka vaativat korjaustoimenpiteitä. Tässä työssä sovelletaan tiedonlouhinnan ja koneoppimisen menetelmiä, erityisesti luokittelua ja klusterointia, auton maalipintoihin liittyvään aineistoon, joka kerätään robotti-avusteisen reflectCONTROL-mittausjärjestelmän avulla. Järjestelmän valmistaja on saksalainen Micro-Epsilon GmbH & Co. KG. Työssä tutkitaan automaattisen virheenluokittelun mahdollisuutta koneoppimista hyödyntäen sekä pyritään keräämään uutta tietoa datasta klusteroinnin avulla.

Työ on jaettu teoreettiseen ja empiiriseen osaan. Teoreettisessa osassa esitellään koneoppimiseen liittyvät peruskäsitteet ja menetelmät, joita hyödynnetään empiirisessä tutkimuksessa. Luokitteluun käytetään kahta eri algoritmia; satunnainen metsä -luokittelijaa ja tukivektorikonetta. Klusterointiin käytetään kolmea eri algoritmia, joita ovat k-means, Affinity Propagation ja HDBSCAN. Lisäksi työssä hyödynnetään kahta eri dimension vähentämismenetelmää, joita ovat pääkomponenttianalyysi ja t-SNE. Myös pistepilvi histogrammi -menetelmää sovelletaan 3D-dataan ja tutkitaan mahdollisuutta hyödyntää sen perusteella laskettuja attribuutteja luokittelussa ja klusteroinnissa. Suurimman haasteen tutkimukselle aiheuttaa käytössä olevan aineiston vinouma eri virheluokkien koon suhteen.

Empiirisen tutkimuksen perusteella satunnainen metsä -luokittelija ja tukivektorikone tuottavat hyvin samankaltaisia luokittelutuloksia sekä binäärisessä että usean luokan tapauksessa. Usean luokan tapauksessa tulokset osoittavat, että käyttämällä tasapainotettua dataa, on mahdollista erottaa eri virheluokat toisistaan. Voidaan myös todeta, että datalla on sisäinen rakenne, joka kuitenkaan ei vastaa ulkoisesti määriteltyä virheluokitusta. Tulokset viittaavat siihen, että pistepilvi histogrammi -menetelmän avulla lasketut attribuutit eivät paranna luokittelutuloksia merkittävästi, mutta klusteroinnissa parantavat sisäisen rakenteen vastaavuutta ulkoisen virheluokituksen suhteen. Nämä tulokset ovat lupaavia jatkon kannalta, sillä niiden perusteella voidaan olettaa, että klusterointia voidaan käyttää ennalta luokittelemattomaan aineistoon. Lisäksi voidaan olettaa, että erityyppisten virheiden erottaminen toisistaan on mahdollista, kunhan aineiston vinouma otetaan huomioon.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Science and Engineering
Pinja Kuosmanen: Data Mining on Deflectometric Data of Surface Defects
Master of Science Thesis, 70 pages, 5 appendix pages
August 2017
Major: Mathematics
Supervisors: Tapio Elomaa, Simo Ali-Löytty
Keywords: surface defect recognition, automated surface inspection, machine learning, data mining

The objective of this thesis is to apply machine learning and data mining methods, especially classification and clustering, onto deflectometric data of surface defects found on car bodies. The data is acquired via robot-assisted automated surface inspection system, manufactured by Micro-Epsilon GmbH & Co. KG, called reflectCONTROL. The measurement method of reflectCONTROL is based on Phase Measuring Deflectometry. The aim is to explore the possibility of automated defect classification via learning algorithms, and to gain new insight about the deflectometric data obtained from surface inspection process via clustering.

This thesis is divided into theoretical part and empirical part. Basic concepts of machine learning and methods used in empirical evaluation are introduced in the theoretical part. The methods include two classification algorithms; Random Forest Classifier and Support Vector Machines. The three clustering algorithms used are k-means, Affinity Propagation and HDBSCAN. Furthermore, dimensionality reduction methods, such as Principal Component Analysis and t-SNE are included. In addition, the possibility of using Point Feature Histograms in the context of deflectometric data and feature generation is explored. The biggest challenge related to this research is that the data set used is highly unbalanced, the biggest class dominating over others in the learning tasks.

The empirical study indicates that Random Forest Classifier and Support Vector Machines perform very similarly in classification tasks. Furthermore, it is possible to distinguish between different classes using multiclass classification on balanced data. It was also found via silhouette analysis and dimensionality reduction, that internal structure exists in the data, but it does not correspond to the human-assigned class labels of the defects. Finally, the study indicates that Point Feature Histogram-based features do not improve the classification performance significantly, but are helpful in clustering tasks by improving the correspondence between internal structure and human-assigned labels. The results of this study are promising for further research, suggesting that it is possible to conduct research on unlabeled data via clustering, and distinguish between different defect classes using appropriately selected data.

FOREWORD

This thesis is a continuation to previous work done by field experts at Micro-Epsilon GmbH, and it is written as completion to my Master's degree in Science and Engineering at Tampere University of Technology. I have been very fortunate to be able to work with this interesting project, and gain experience and skills that I had never imagined. This thesis has also been very important for me, as it has confirmed that the field of data science is something I would like to build my future career on. Most importantly, I have enjoyed every minute of this journey. This is due to having the best possible support during the entire project, thanks to awesome people at Micro-Epsilon, especially Mr. Reiner Kickingereder, who has made it possible for me to grow as a professional. I would like to sincerely thank Mr. Thomas Wisspeintner and Micro-Epsilon for giving me this opportunity and for all the hospitality during my visits in Ortenburg, Germany.

This project would not have been the same without my supervisors at TUT. I would like to thank Professor Tapio Elomaa and University Lecturer Simo Ali-Löytty for finding the time to supervise this thesis and for arranging peaceful surroundings for me to work in. It has been invaluable to have multiple advisers and versatile views of the topic. I would also like to thank Mrs. Liisa Kuulasmaa, my mathematics teacher at junior high school, who was the first person to teach me the fascination of mathematics as well as the importance of hard work. At that time, making me do something that felt like endless number of assignments, later proved to be the vital and necessary foundation to succeed in my studies.

Last, I would like to tank my family, father Jouko, mother Minna-Kaisa and brother Patrik, for endless love, support, and encouragement. I have always been able to make my own decisions and choose my own paths, but sometimes a tiny push to the right direction at the right moment has been just what I needed to reach this point. Thanks to my family, the attitude towards the importance of education, and the urge of being the best version of myself have always been something to consider self-evident. I would also like to thank my friends, who have been there for me during my studies in Helsinki and Tampere, you have made these years worthwhile.

Tampere, 22nd of August 2017

Pinja Kuosmanen

CONTENTS

LIST OF ANNOTATIONS AND ABBREVIATIONS

AI	Artificial Intelligence
AP	Affinity Propagation
PCA	Principal Component Analysis
PFH	Point Feature Histogram
PMD	Phase Measuring Deflectormetry
SVC	Support Vector Classification
SVM	Support Vector Machine
t-SNE	t-Distributed Stochastic Neighbor Embedding
$a(i)$	Intra-cluster mean distance for instance $\mathbf{x}_i$
$acc(y, \hat{y})$	Fraction of correct predictions in classification
$ava(i, j)$	Availability in Affinity Propagation
AMI	Adjusted Mutual Information
ARI	Adjusted Rand Index
$b(i)$	nearest inter-cluster mean distance for instance $\mathbf{x}_i$
$\mathbf{C}$	Confusion matrix
C	Ground truth class labeling
$Cost_{\mathrm{t-SNE}}$	Cost function of t-SNE algorithm
d	Dimension on an instance
δ	Any distance metric between two points
δ_E	Euclidean distance between two points
$EmpLoss_{0/1,E}$	Empirical loss of the set E of examples
E	Set of all possible input-output pairs (examples)
E	Expected value
$\mathbf{e}$	Vector of all ones
$\mathrm{F_{og}}$	Original features
$\mathrm{F_{gauss}}$	Gaussian features
$\mathrm{F_{PFH}}$	PFH features
F1	Original and gaussian features
F2	Combination of best features
G	Cluster labeling
$GenLoss_{0/1}$	Generalization loss of a hypothesis
$h : X \to Y$	Hypothesis

$h^\star$	Best hypothesis
$\hat{h}^\star$	Estimated best hypothesis
$H(C)$	Entropy of the classes in C
$H(C \mid G)$	Conditional entropy of the classes given the cluster assignments
$\mathcal{H}$	Hypothesis class
K	Kernel function
k	Number of clusters (or number of folds in cross-validation)
KL	Kullback-Leibler divergence
$\mathbf{S}$	Covariance matrix
$L_{0/1}$	Zero-one loss of a hypothesis
l	Number of pairs of instances that are in same cluster and same class
m	Number of pairs of instances that are in different cluster and same class
MI	Mutual Information
n	Number of examples
p_i	Point of a three dimensional surface
$P(X, Y)$	Prior probability distribution over the set of examples
R	Regularization parameter
RI	Rand Index
$res(i, k)$	Responsibility in Affinity Propagation
$\langle \mathbf{x}, y \rangle$	Training example
$\mathbf{x}$	Instance
$x_1, \cdots, x_d$	Features / attributes
X_i	Range of an attribute
X	Instance space
y	Output variable
$\mathbf{y}$	Vector of all output variables
Y	Set of output variables
c	Completeness
s	Silhouette score of a clustering
$s(i)$	Silhouette Coefficient of an instance $\mathbf{x}_i$
$sim(i, j)$	Similarity between points in Affinity Propagation
h	Homogeneity

max_depth	Maximum depth of the tree for Random Forest
max_features	Size of random subsets of features in Random Forest
min_samples_split	Minimum number of samples required to split an internal node in Random Forest
n_estimators	The number of trees in Random Forest

1. INTRODUCTION

Automatic surface inspection is an important part of quality control in several manufacturing fields, including automotive industry. Painted car bodies usually have small defects in the paint coat, which require detection and possibly reworking. The paint is meant to protect the surface, but perfectly painted surface is also an emotional measure of quality for the customer. The detection of these defects performed by human experts is time-consuming, costly, and subjective. Hence, automated surface inspection is desirable for manufacturing companies. There are several different defect types, that can occur on the painted surface, e.g. inclusions, craters, dents, condensates, and paint runs. These defects can occur in the top coat layer, but also below the top coat, in the lower paint layers.

One possible method to measure reflective surfaces, such as painted car bodies, is Phase Measuring Deflectormetry. This method is used in the surface inspection by Micro-Epsilon GmbH & Co. KG. The robot-assisted system based on Phase Measuring Deflectormetry is called reflectCONTROL, and it already offers detection of all relevant defect types based on defect size. However, defects of the same size can actually belong to separate categories, which poses a challenge to the automation of defect recognition. So far the recognition of different kinds of defects is only performed by human experts. In this thesis, machine learning algorithms are applied to the deflectometric data acquired from the surface inspection process of reflectCONTROL. The objective is to explore the possibility of automated defect classification via supervised and unsupervised learning. Different algorithms for clustering and classification tasks are applied on the data, and the performance of these algorithms are compared in order to see if distinct algorithms perform similarly on these tasks.

The two methods chosen for classification task are Random Forest Classifier and Support Vector Classifier. Random Forests are decision tree based ensemble learning method. Support Vector Classifier is a learning method based on finding the best separating decision boundaries between distinct classes, which involves optimization of a loss function. For clustering, the methods chosen are k-means, Affinity Propagation, and HDBSCAN, which all are very different from each other. k-means is a very traditional clustering algorithm, which uses inertia as minimization criteria. Inertia can be seen as the measure of internal coherence. Affinity Propagation is a clustering method based on passing messages between data points by viewing each

data point as a node in a network. The messages are then recursively transmitted along the edges of the network until a good set of cluster centers or exemplars is obtained. HDBSCAN, in turn, is a density-based and hierarchical method extending DBSCAN. Both Affinity Propagation and HDBSCAN do not require the number of clusters to be determined before running the algorithm whereas k-means does. Clustering results are evaluated for different numbers of clusters with each algorithm.

One part of the research done for this thesis is exploring the possibility of using Point Feature Histograms to create more features for classification and clustering tasks, as well as exploring whether these features are useful in the context of deflectometric data. Point Feature Histograms are a statistical representation of three-dimensional shapes, and they are commonly applied in 3D object recognition. Classification and clustering are first performed on the deflectometric data without these features, and then repeated using the features obtained based on PFH descriptors in order to compare the results with the original ones. Based on the findings of this thesis, the performance of Point Feature Histogram descriptors on deflectometric data is evaluated.

In Chapter 2, theoretical background related to Phase Measuring Deflectormetry and reflectCONTROL is introduced. Furthermore, related work in the field of automatic surface inspection and defect recognition is briefly overviewed in order to give the reader an impression about the previous work done in the field. In addition, the basic concepts of machine learning, classification, and clustering are defined together with the notations used in this thesis. A small example is given to clarify these notations. Finally, the metrics used in empirical evaluation are explained in more detail.

In Chapter 3, the learning algorithms used in empirical evaluation are first introduced. Two dimensionality reduction techniques, Principal Component Analysis and t-SNE, are also explained together with Point Feature Histograms and related descriptors. The data set and its features are as well as preprocessing steps also explained. Finally, three research questions are determined and before going into results, the steps taken in empirical evaluation in order to answer to the research questions are described.

In Chapter 4, results of empirical evaluation are presented. The results are divided into classification and clustering tasks, and in more detail, into results obtained with different algorithms. Finally, some discussion about the different results as well as answers to some of the research questions are given. In Chapter 5, the research questions are then answered in detail. After answering to the research questions, some suggestions for future work are discussed.

2. THEORETICAL BACKGROUND AND PREMISE

In this chapter, the theoretical background related to Phase Measuring Deflectormetry together with its implementation by Micro-Epsilon GmbH & Co. KG are introduced. In addition, related work in the field of automatic surface inspection and defect recognition is briefly overviewed. In the last section, the basic concepts and notations related to machine learning, classification, and clustering are defined and illustrated via small example data set. For both classification and clustering tasks, the metrics used in empirical evaluation are explained in more detail.

2.1 Phase Measuring Deflectormetry

Phase Measuring Deflectormetry (PMD) was introduced by Knauer et al. as a method to measure the shape of specular free-form surfaces [15]. The basic principle is to reflect and phase shift sinusoidal patterns on the inspected surface, and to observe the patterns reflected via the surface with a camera setting [15]. The challenge in measuring reflective free-form surfaces is that the surfaces and structures on the surfaces are not really visible, but only their effect can be detected through the reflection of incoming light. This is because any structures on the surface lead to distortions of the reflected patterns. These disortions can then be detected and evaluated by a software, and image processing algorithms can be used to reconstruct the surface. The basic setup used in Phase Measuring Deflectometry is described in Figure 2.1.

reflectCONTROL

reflectCONTROL from Micro-Epsilon GmbH & Co. KG is a robot-based surface inspection system for automotive and other industries based on phase measuring deflectometry. The objective is to allow fast, fully automated in-line inspection with high detection rate for all defect types. Currently, the surface inspection system already offers detection of all relevant defect types as well as optional automated marking of found defects based on defect size. Using image processing algorithms, measurement data can be converted into three channels: local curvature, reflectivity, and base intensity. These channels are then evaluated in order to detect anomalies

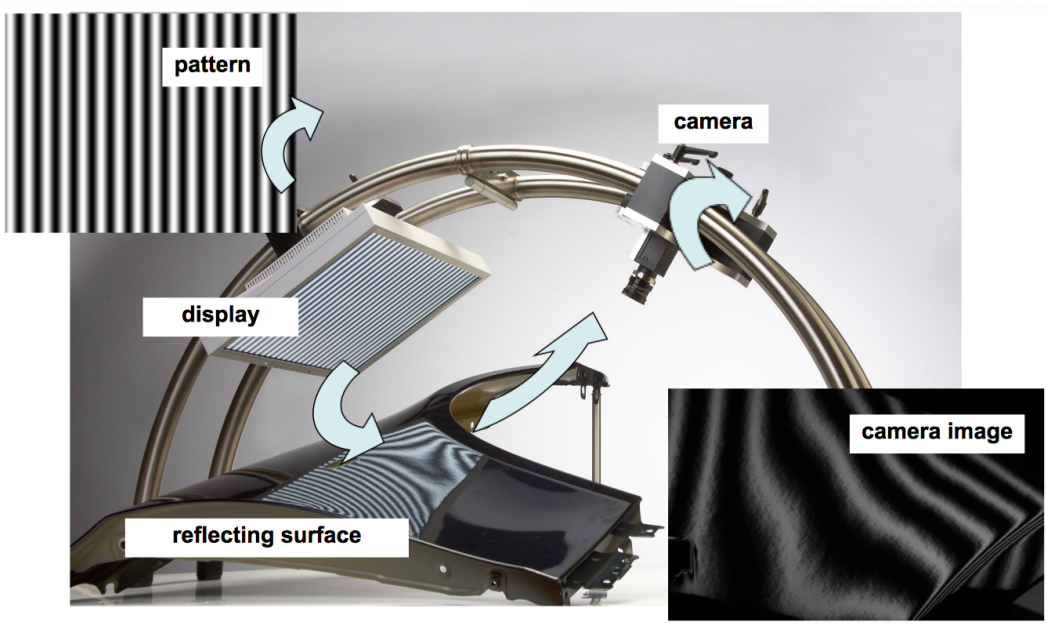

Figure 2.1: Measurement principle of Phase Measuring Deflectometry (courtesy of Micro-Espilon).

(defects) and to allow 3D-reconstruction of the recognized defects. However, the automated classification is limited to categorization based on defect size, while defects of the same size can actually belong to separate categories, e.g. inclusions, craters, dents, and paint runs. So far the recognition and separation of different kind of defects from another can only be performed by human inspectors. The robot setting of reflectCONTROL can be seen in Figure 2.2. The objective of this thesis is to implement machine learning methods, especially classification and clustering, to gain new information about the structure of the data obtained from described surface inspection process (reflection data), and to explore the possibility of automated defect classification via supervised learning.

2.2 Related work

Automatic surface inspection is an important part of quality control in several manufacturing fields. Manual surface inspection is subjective, repetitive process, and prone to errors as it is performed by human experts. Automation is often desirable in order to replace exessive manual work and improve accuracy of defect detection. Hence, there has been many efforts in the past decades to improve the surface inspection process and automation. According to Karbacher et al., the small defect detection problem on shiny surfaces was first solved by detecting curvature changes from reflected light [14]. These kind of approaches are usually known as deflectometric approaches. The challenge with 2D methods is that the reflection depends

Figure 2.2: The robot setting of reflectCONTROL. Photo provided by Micro-Epsilon.

on the local orientation of the surface [14].

In 1999, Karbacher et al. demonstrated a method to visualize small defects on car bodies using 3D sensors and signal processing [14]. They also developed a case-based reasoning algorithm for a German car company for defect analysis and classification. The algorithm decides whether the candidate really is a defect and furthermore, whether it is relevant. The motivation was to simplify the repairing process by detecting the defects before applying varnish on top.

In 2000, Lilienblum et al. described a method for an automatic detection of small dents in car bodies [17] combining 3D optical measurement system with neural networks. The 3D system used in [17] is a photogrammetrical method with improved spatial resolution. For automatic defect recognition, Lilienbaum et al. use neural networks by calculating the difference between master and measured workpiece. The limitations of their method follow from high requirements for the adjustment of master and measured workpiece [17]. According to the empirical measurements, an automatic detection of dents with a height of 20μm is possible with the method.

In 2004, Jia et al. described a real-time visual inspection (machine vision) system for steel manufacturing quality control, using SVMs to automatically learn defect patterns [11]. Based on experimental results on image data from hot rolling manufacturing, they found the proposed system to be effective in detecting steel surface defects; the speed of the system being less than 6 ms per one megabyte image. They

furthermore conclude that experimental results demonstrate the potential of SVMs as promising classifiers for defect detection in real-time manufacturing environment.

In 2006, Döring et al. continued their previous work in the field of automatic quality assessment of car body parts in order to achieve a higher reliability of the defect detection [4]. They show that more accurate classification models can be built on the basis of a preprocessed, more consistent training set. They use partially supervised learning strategy to improve previous classification accuracy on unbalanced data set, and demonstrate improvement on average test accuracy between 2.93% and 5.88%, depending on the used method. In the experiments with unmodified data, they achieved the average classification accuracy of 74.97% on the training data, and 76,56% on the test data. For classification they implement decision trees, fuzzy decision trees, NEFCLASS, and mixed fuzzy rules, from which the decision trees performed with best average accuracy. They also combine unsupervised classification (clustering) with supervised methods.

In 2009, Jones and Aoun extended the work done by Gary Bradski at Willow Garage on point cloud based object recognition by applying a machine learning approach [12]. They demonstrated how histograms calculated from 3D point clouds can be used in a machine learning algorithm in object recognition and classification. In their experiments, they used histograms based on normal vectors, similar to ones proposed by Rusu et al. in [31] (discussed later), together with SVM classification. Their experiment data consists of 3D point clouds of IKEA models with 40 examples of each model. They concluded that combining point cloud histograms with machine learning algorithms is a good approach for solving the multi-class classification problem on a point cloud data.

In 2011, Kamani et al. proposed a new computer vision system for automatic painted car body inspection in the context of quality control [13]. They combined computer vision with Bayesian classifier in order to detect predict the type of the detected defect. They used both synthetic and real defects on their experiments, showing that the proposed system achieved a high defect detection and classification rate, average accuracy reaching 96,2% for six different defect types.

2.3 Machine Learning

Machine learning has evolved out of artificial intelligence (AI) and since then has formed a dominant subfield concerned with creating algorithms and computer systems so that a machine can learn by accumulating experience through data, systems, and programs [10]. Thus, the goal of machine learning is to improve machine's performance over time on specific tasks by accumulating experience. The general idea behind machine learning is such, that the computer learns to perform task by studying a training set of examples [18].

The most well-known types of machine learning problems are problems involving classification and clustering. Today machine learning is applied to a range of domains including security heuristics, image analysis, deep learning, object recognition, and pattern recognition [18]. It is also common in the machine learning community to divide learning problems into various categories, two of which are supervised learning and unsupervised learning. Supervised learning is a machine learning term for predictive data mining, whereas unsupervised learning is also known as descriptive data mining [10]. In fact, most methods used in data mining are related to methods developed in machine learning [10]. In this thesis, both descriptive and predictive data mining methods are applied, since it is of interest to find inherent structures and trends as well as build models for classification task.

2.3.1 Supervised learning and classification

In the supervised machine learning setting, an *example* is a tuple $\langle \mathbf{x}, y \rangle$, where the d-dimensional *instance* $\mathbf{x} = \langle x_1, \ldots, x_d \rangle$ records the values of features (attributes) $x_i \in X_i$, $i = 1, \ldots, d$ and y is the corresponding output variable [22]. Let n denote the number of examples. The range of an attribute as well as the output can be either categorical or continuous: E.g., $X_1 = \{\text{red}, \text{green}, \text{blue}\}$ and $X_2 = \mathbb{R}$. Categorical output variable yields a *classification problem* whereas continuous one yields a *regression problem* [10]. Classification problems are also divided into *binary classification* and *multiclass classification* based on the number of possible output categories [22]. In binary case, there are only two possible output values, e.g. $Y = \{1, -1\}$, whereas in multiclass case, there are more than two possible output values, one of which each instance is then being assigned: e.g: $Y = \{\text{red}, \text{green}, \text{blue}, \text{green}\}$.

Let $X = X_1 \times \ldots \times X_d$ be the *instance space*. The examples are then sampled from the application domain based on a *prior probability distribution* $P(X, Y)$ [27]. In classification one aims at learning a *hypothesis* $h : X \to Y$. The hypothesis can be used to predict the class of a previously unseen instance $\mathbf{x}$. The goodness of a hypothesis is measured by its accuracy. Let us formulate accuracy through the *zero-one loss* $L_{0/1}$ of a hypothesis. The following formulation is adapted from [22] and [27].

$$L_{0/1}(y, \hat{y}) = \begin{cases} 0, & \text{if} \quad y = \hat{y} \\ 1, & \text{otherwise.} \end{cases} \tag{2.1}$$

Zero-one loss counts all the erroneous predictions with the same weight, but in some situations it is more convenient to formulate such loss function, that takes the gravity of an error into account. Let E denote all possible input-output examples.

The expected generalization loss for a hypothesis h is

$$GenLoss_{0/1}(h) = \sum_{\langle \mathbf{x}, y \rangle \in E} L(y, h(\mathbf{x}))P(\mathbf{x}, y). \tag{2.2}$$

The best hypothesis $h^\star$ is the one with the minimum expected generalization loss from a fixed *hypothesis class* $\mathcal{H}$.

$$h^\star = \arg\min_{h \in \mathcal{H}} GenLoss_{0/1}(h). \tag{2.3}$$

However, $P(\mathbf{x}, y)$ is usually unknown and, hence, the learner can only estimate generalization loss with *empirical loss* on the set of examples E:

$$EmpLoss_{0/1,E}(h) = \frac{1}{n} \sum_{\langle \mathbf{x}, y \rangle \in E} L_{0/1}(y, h(\mathbf{x})). \tag{2.4}$$

The estimated best hypothesis $\hat{h}^\star$ is then

$$\hat{h}^\star = \arg\min_{h \in \mathcal{H}} EmpLoss_{0/1,E}(h). \tag{2.5}$$

Finally, the accuracy of a hypothesis can be formulated using empirical loss:

$$acc(y, \hat{y}) = 1 - EmpLoss_{0/1}. \tag{2.6}$$

To clarify these notations, a brief example is given using Teaching Assistant Evaluation Data (TAE data) (Figure 2.3) from UCI Machine Learning Repository [16]. Figure 2.3 contains only a small subset of the entire TAE data set.

Example 2.3.1. TAE data set of $n = 20$ examples consists of five categorical attributes and one numerical attribute together with categorical output variable "Class", which refers to evaluation of teaching performance. The possible scores are 1 = Low, 2 = Medium, 3 = High. Hence, $Y = \{1, 2, 3\}$, and each row of this data set corresponds to one example $\langle \mathbf{x}_i, y_i \rangle$, where $i = 0, \ldots, 19$. Each $\mathbf{x}_i$ records the values of six attributes: "Native English Speaker", "Course Instructor", "Course", "Summer or Regular semester", and "Class size". Thus, each $\mathbf{x}_i$ is a 6-dimensional instance. Note that, for example, "Class size" is a numerical attribute, whereas "Summer or Regular semester" is a binary categorical attribute and hence, $X_{\text{ClassSize}} = \mathbb{N}$ whereas $X_{\text{SummerOrRegular}} = \{1, 2\}$. Using the notations introduced above, it is now possible to write the example located on the first row of the data set in Figure 2.3 as $\langle \mathbf{x}_0, y_0 \rangle$, where the instance is $\mathbf{x}_0 = \langle 1, 23, 3, 1, 19 \rangle$ and the corresponding output is $y_0 = 3$. ◊

	Native English speaker	Course instructor	Course	Summer or regular semester	Class size	Class
0	1	23	3	1	19	3
1	2	15	3	1	17	3
2	1	23	3	2	49	3
3	1	5	2	2	33	3
4	2	7	11	2	55	3
5	2	23	3	1	20	3
6	2	9	5	2	19	3
7	2	10	3	2	27	3
8	1	22	3	1	58	3
9	2	15	3	1	20	3
10	2	10	22	2	9	3
11	2	13	1	2	30	3
12	2	18	21	2	29	3
13	2	6	17	2	39	3
14	2	6	17	2	42	2
15	2	6	17	2	43	2
16	2	7	11	2	10	2
17	2	22	3	2	46	2
18	2	13	3	1	10	2
19	2	7	25	2	42	2

Figure 2.3: Toy example from UCI Machine Learning Repository [16] called Teaching Assistant Evaluation Data Set (TAE data).

Classification

The general idea of classification is to find a way to assign a future example to one of predefined categories based on previous experience. Hence, classification is based upon measurements on that future example together with the experience obtained from learning instances of similar example. The different approaches of classification can be divided into statistical and machine learning approaches: Statistical approaches are characterised by an assumption of an underlying probability model out of which the probability of an example being in each class is deriven. Machine learning approaches, on the other hand, are generally concerned as procedures where the classification results from a sequence of logical steps [20]. These steps are usually performed via different classification algorithms, such as logical regression, classification trees, support vector machines (SVM), random forests, or artificial neural networks (ANNs) [18]. The algorithm aims to find the model that best fits the relationship between the attribute set and class label of the input data [32].

The challenge in the classification task is to generate such model, that both fits

the input data well but also correctly predicts the class labels of completely new, independent instances. In other words, a good model is such that it has low *training error* as well as low *generalization error* [32]. Intuitively it makes sense to try to find the model that fits the training data as accurately as possible. However, if the model fits the training data too well, it usually makes the generated model too complicated and results in very small learning error but very large generalization error [10]. This situation is called *overfitting*. In contrast, *underfitting* can occur if the model is too simple and has not learned the actual structure of the data. Underfitting results in poor performance for both training and generalization errors.

Sometimes finding new independent data for testing the generated model may be hard or even impossible. It is a common practice to divide the data readily available into a *training set* and a *test set*, assumed that the two sets are generated by the same underlying distribution [10]. The test set is held back from the whole process of model generation and introduced only afterwards for assessment. Using only a part of the entire data for model generation may seem like suboptimal solution as some relevant information might be lost. Alternative and popular data-splitting method, *k-fold cross-validation* (discussed later) is often used to address this challenge of having enough information for both training and testing. Although this is a computationally intensive technique, it uses the entire data in a more efficient manner than the plain division into a learning set and an independent test set [10].

2.3.2 Classification performance evaluation

In order to assess the goodness and fit of an obtained classification model, some performance evaluation metrics are needed. The commonly used evaluation metrics include accuracy, cross-validation score, and confusion matrix.

Accuracy Score

Accuracy is one of the most intuitive and easily interpreted metrics for classification models. The typical measure of prediction accuracy is based on empirical loss which, for classfication task, is the propability of missclassifying an instance [10]. In this thesis, the implementation of accuracy score from Scikit-learn library is applied. The implementation of accuracy score in Scikit-learn computes by default the fraction of correct predictions [24]. In multilabel classification, the function gives the subset accuracy. If the entire set of predicted labels $\hat{y}$ match with the set of true labels y, then the subset accuracy is 1.0 and otherwise 0.0 [24].

Definition 2.3.1 (Accuracy)**.** Let $\langle \mathbf{x}_i, y_i \rangle$ be the ith example and $h(\mathbf{x}_i) = \hat{y}_i$ the predicted value, and S the sample with n examples. Then the fraction of the correct predictions over n is defined as:

$$acc(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} 1 - L_{0/1}(\hat{y}_i, y_i). \tag{2.7}$$

△

Cross-Validation Score

Accuracy score is a simple and straightforward way to measure goodness and fit of the generated model. It is not, however, sufficient on its own. As stated in [24], learning the prediction function and testing it only on the same data is a methodological mistake, since this would result in overfitting the model and failing to predict anything on fresh data. To avoid overfitting, it is a common practice to evaluate generalization via cross-validation score. The procedure adapted from [24] is the following:

- The entire data is randomly divided into k non-overlapping groups of roughly equal size
- A model is trained using $k - 1$ of the folds as training data;
- the resulting model is validated on the remaining part of the data, i.e., the omitted group is used as a test set to compute a performance measure such as accuracy.

The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop k times, each time removing a different group. The basic partitioning principle is illustrated in Figure 2.4. Note that later on in this thesis, k always denotes the number of clusters in a clustering assignment. Here, however, k denotes the number of folds in cross-validation as it is the established notation.

Confusion Matrix

Confusion matrix is a common technique to evaluate and summarize the perormance of classification algorithm, and to form even better understanding of the results obtained. Instead of displaying a single value of accuracy, confusion matrix shows the number of incorrect predictions; the way in which the model is confused. Accuracy alone can be misleading in the situation where the classes are of different sizes. Hence, confusion matrix can be very useful in the context of reflection data. The definition provided in [24] is the following:

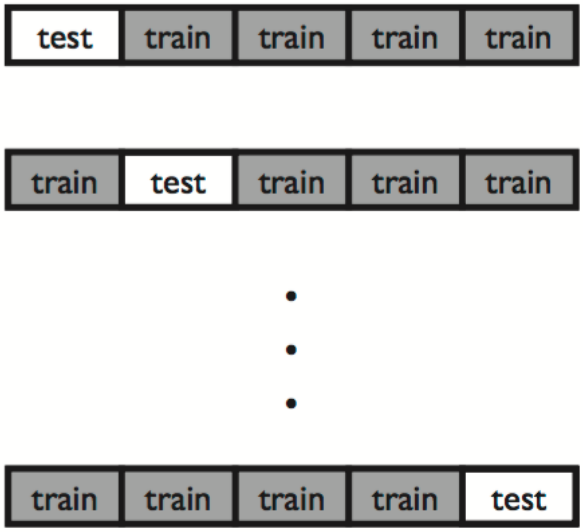

Figure 2.4: Illustration of basic cross-validation principle with 5 folds. Figure from [22].

Definition 2.3.2 (Confusion matrix)**.** A confusion matrix $\mathbf{C}$ is such that $\mathbf{C}_{i,j}$ is equal to the number of observations known to be in group i, but predicted to be in group j. In a 2×2 confusion matrix, the outcomes can be formulated as follows:

$$\mathbf{C} = \begin{bmatrix} \text{TruePositives} & \text{FalseNegatives} \\ \text{FalsePositives} & \text{TrueNegatives} \end{bmatrix}$$

△

2.3.3 Unsupervised learning and clustering

In contrast to the supervised learning, unsupervised learning problems are defined to be those with no information available of the correct output variable y. Thus, the goal of unsupervised learning differs from that of supervised learning: In supervised learning the perspective is to study the relationships of input and output variables whereas in unsupervised learning it is to explore the characteristics of the input variables only. The input variables can be described as a set of patterns and the output as the underlying motifs that generated the patterns [18]. Tasks related to unsupervised learning commonly involve joint probability estimation, clustering, locating outliers or imputing missing data.

Clustering

Clustering or cluster analysis can be defined as a statistical tool for arranging data into natural groups [10]. Clustering resembles classification but the fundamental difference between those two is that in clustering the number of groups (clusters), denoted by k, is unknown. The discovered grouping of n instances, denoted by $G = \{G_1, \ldots, G_k\}$, where each $G_i, i = 1, \ldots, k$ corresponds to a single cluster, is only based on the information found in the data without any externally obtained labeling. In fact, clustering is sometimes referred to as unsupervised classification, since it can be seen as a form of classification that derives the labeling only from the data [32].

Cluster analysis tasks can be divided into clustering for understanding data and clustering for utility [32]. In the context of understanding data, cluster analysis is the study for automatically finding classes, whereas in the context of utility, cluster analysis can be seen as the study of finding the most representative cluster prototypes [32]. The principle behind clustering algorithms is such that the algorithms take as input a dataset of various dimensions and partition it into clusters satisfying certain criteria [18]. There is no single formal definition for a cluster that would be sufficient for every situation. The notion of a cluster is usually not well defined and depends vastly on nature of data, desired results, and the clustering method or algorithm chosen for the task [32]. In fact, it is argued that the definitions of a cluster in the literature reflect the different philosophical points on the topic [5].

While clustering tasks can be divided according to the purpose of the task, clustering methods can also be divided into various groups. Tan et al. divide clustering methods into hierarchical, partitional, exclusive, overlapping, and fuzzy methods: *Partitional clustering* methods group data into non-overlapping clusters such that each data object is in exactly one cluster, and if clusters can have subclusters, then a *hierarchical clustering* is obtained. *Exclusive clustering* assigns every object of the data to exactly one cluster, while non-exclusive or *overlapping clustering* allows an object to simultaneously belong to more than just one cluster. In a *fuzzy (or soft) clustering*, on the other hand, every object is assigned to every cluster with membership weight between 0 and 1, and hence, the clusters are treated as fuzzy sets. A complete clustering assign every object to at least one cluster, while a partial clustering does not. In partial clustering, the objects that are not assigned to any cluster can be seen as outliers or noise [32]. Popular algorithms include methods such as k-means, Affinity propagation, Mean Shift, Spectral Clustering, DBScan together with it's more recent version, HDBScan.

2.3.4 Clustering evaluation

The most challenging part of cluster analysis is the evaluation of obtained clustering and interpreting the results, since it is not as straightforward as classification evaluation. As stated in [32], cluster evaluation is not well-developed or commonly used part of cluster analysis. Even the concept of a good clustering depends on the application. There are, however, some well established methods for cluster evaluation, which include *internal* as well as *external* metrics. Internal metrics are used to evaluate only the goodness of the model itself, and do not require any information about the *groud truth labeling* C (the knowledge of externally defined true labels), whereas external metrics are used to evaluate the goodness of the model with respect to the ground truth labeling. The clustering evaluation metrics used in this thesis are silhouette coefficient, homogeneity score, completeness score, Adjusted Rand Index (ARI), and finally, Adjusted Mutual Information (AMI).

Silhouette Coefficient

The silhouette coefficient is a metric for internal evaluation and hence, using only the model itself for evaluation process. The score is calculated using mean intra-cluster distance and the mean nearest-cluster distance for each instance [24]. Higher scores (near 1.0) for silhouette coefficient are related to better defined clusters whereas negative values correspond to a case where the average distance of points within the cluster is greater than the minimum distance to points in another cluster, which generally indicates that an instance has been assigned to a wrong cluster [32], [26]. Scores near zero generally indicate overlapping clusters [24]. The formulation based on [26] is the following:

Definition 2.3.3 (Silhouette Coefficient)**.** Assume any partition (clustering) G and any distance metric δ. (The default being euclidean distance in Scikit-learn library's implementation). Take any instance $\mathbf{x}_i$ from the data set and denote by G_1 the cluster to which it is assigned. Let $a(i)$, $\delta(i, G_i)$ and $b(i)$ be the following scores:

- $a(i)$ is the mean distance between an instance $\mathbf{x}_i$ and all other instances in the same cluster G_1.
- $\delta(i, G_j)$ is the mean distance between $\mathbf{x}_i$ and all instances of any other cluster G_j, $j \neq 1$.
- $b(i) = \min_{j \neq 1} \delta(i, G_j)$ is the mean distance between instance $\mathbf{x}_i$ and all other instances in the next nearest cluster.

The silhouette coefficient $s(i)$ for a single instance $\mathbf{x}_i$ is then given by:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}. \tag{2.8}$$

△

In Figure 2.5, $a(i)$ is represented by the mean length of all lines inside cluster G_1 and $b(i)$ is represented by the mean length of all lines going from $\mathbf{x}_i$ to G_2, which is the next nearest cluster.

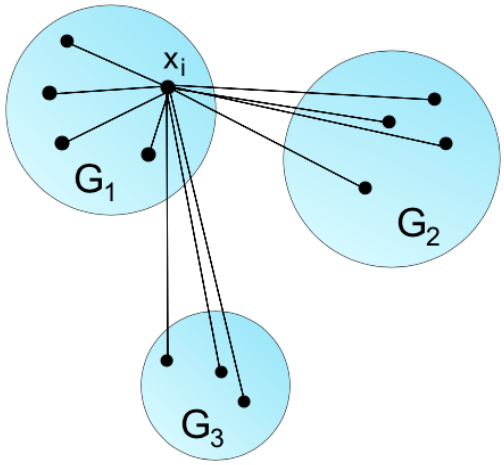

Figure 2.5: An illustration of the elements related to silhouette coefficient. Adapted from [26].

The *silhouette score* for the entire clustering (set of instances), denoted by s, is then given as a mean of the silhouette coefficients for each instance. Silhouette score is a useful metric for evaluating the internal goodness of the clustering obtained, but does not give any information about the cluster labeling compared to the ground truth labeling.

Homogeneity and Completeness

In addition to internal evaluation, it is also possible to compare the obtained cluster labeling to the ground truth labeling, assuming such knowledge exists. For that purpose, external evaluation metrics are applied. Given the ground truth labeling C, two relatively intuitive metrics for external evaluation of clustering goodness are homogeneity and completeness. These both are objectives, that are desirable for any cluster assignment, defined originally in [25], and summarized in [24].

Let us first define the *entropy* $H(C)$ of the classes in C. Recall that n is the total number of examples and let n_{C_i} be the number of examples belonging to class C_i.

Entropy is now given by:

$$H(C) = -\sum_{i=1}^{|C|} \frac{n_{C_i}}{n} \cdot \log_2 \left(\frac{n_{C_i}}{n} \right). \tag{2.9}$$

Let n_{G_i} be the number of examples belonging to cluster G_i and n_{C_i,G_j} the number of examples from class C_i assigned to cluster G_j. The *conditional entropy* $H(C \mid G)$ of the classes given the cluster assignments is given by:

$$H(C \mid G) = -\sum_{i=1}^{|C|} \sum_{j=1}^{|G|} \frac{n_{C_i,G_j}}{n} \cdot \log_2 \left(\frac{n_{C_i,G_j}}{n_{G_i}} \right). \tag{2.10}$$

Using equations 2.9 and 2.10, homogeneity and completeness are now defined as follows:

Definition 2.3.4 (Homogeneity)**.** A clustering G satisfies homogeneity if all of its clusters $G_i, i = 1, \ldots, k$ contain only data points which are members of a single class of the ground truth labeling C. Formally:

$$h = 1 - \frac{H(C \mid G)}{H(C)}. \tag{2.11}$$

△

Definition 2.3.5 (Completeness)**.** A clustering satisfies completeness if all the examples that are members of a given class are elements of the same cluster. Formally:

$$c = 1 - \frac{H(G \mid C)}{H(G)}. \tag{2.12}$$

△

Both of these metrics are independent of the absolute values of the labels and hence, a permutation of the class labeling will not change the score. Neither of these metrics are symmetric in the sense that switching true labels with predicted values will, in general, return different scores [24].

Adjusted Rand Index

Adjusted Rand Index (ARI) is a commonly used, symmetric measure for consensus between two labelings. It was first suggested by Hubert and Arabie in [9]. Given

the knowledge of ground truth labels and the cluster labels of the same examples assigned by the clustering algorithm, the Adjusted Rand Index is a function that measures similarity of two assignments, ignoring permutations and with chance normalization [24]. The following mathematical formulation is adapted from [24].

Definition 2.3.6 (Rand Index)**.** Let C be the ground truth labeling and G a clustering, l and m are defined as:

- l is the number of pairs of elements that are in the same class in C and in the same cluster in G,
- m is the number of pairs of elements that are in different classes in C and in different clusters in G.

The raw (unadjusted) Rand Index is then given by:

$$\mathrm{RI} = \frac{l + m}{\binom{n}{2}}, \tag{2.13}$$

where $\binom{n}{2}$ is the total number of possible pairs in the data set. △

The drawback of RI score is that it does not take the chance into account, i.e., it does not guarantee that random label assignments will get values close to zero. To address this, it is beneficial to discount the expected RI of random labelings by defining the Adjusted Rand Index with chance normalization. The formulation of expected RI, E(RI), is described in more detail by Hubert and Arabie in [9].

Definition 2.3.7 (Adjusted Rand Index)**.**

$$\mathrm{ARI} = \frac{\mathrm{RI} - \mathrm{E(RI)}}{\max(\mathrm{RI}) - \mathrm{E(RI)}}. \tag{2.14}$$

△

Adjusted Mutual Information

Given the ground truth labels and assignments based on clustering, the Mutual Information (MI) is a function that measures the agreement of the two assignments ignoring permutations [24]. Two different normalized versions of this metric are available in the Scikit-learn library [24]. In this thesis, Adjusted Mutual Information

(AMI) is chosen for a measure of agreement from these options, since it is normalized against chance. The following formulation is adapted from [35] and [24].
Mutual Information between G and C can be defined by using again equations 2.9 and 2.10:

$$\mathrm{MI}(G, C) = H(G) - H(G \mid C). \tag{2.15}$$

The expected value for the Mutual Information, E(MI), can be calculated using the equation described in more detail by Vinh et al. in [35]. Using the expected value of MI, the Adjusted Mutual Information is defined as follows:

Definition 2.3.8 (Adjusted Mutual Information)**.**

$$\mathrm{AMI} = \frac{\mathrm{MI} - \mathrm{E}(\mathrm{MI})}{\max(H(G), H(C)) - \mathrm{E}(\mathrm{MI})}. \tag{2.16}$$

△

3. RESEARCH METHODOLOGY

3.1 Learning algorithms

In this chapter, the algorithms used in empirical evaluation are first introduced. The algorithms used in classification include Support Vector Machines (SVM) and Random Forest Classifier. The clustering methods include three very different algorithms; k-means, Affinity Propagation, and HDBSCAN. Two dimensionality reduction techniques, Principal Component Analysis and t-SNE, are also briefly explained. One part of the research done for this thesis is exploring the possibility of using Point Feature Histograms to create more features for classification and clustering tasks and whether these features are useful in the context of deflectometric data of surface defects. Hence, Point Feature Histograms and related descriptors are also explained in this chapter. Finally, three research questions are determined and the data sets used in this thesis are described together with the steps taken in empirical evaluation in order to answer the research questions.

3.1.1 Classification algorithms

Support Vector Machines

Support Vector Machines (SVM) is a popular method developed for binary and multiclass supervised learning problems. The objective of SVM is to find the best separating decision boundaries between points that belong to separate classes. SVMs have been successfully applied to classification problems such as handwritten digit recognition and text categorization, and the method is available in both linear and nonlinear versions [10]. SVM involves optimization of a convex loss function under given constraints and hence, is unaffected by problems of local minima [10]. Unlike traditional classifiers, which minimize the empirical loss function, SMVs aim to minimize the upper bound of the generalization loss [11]. In order to deal with nonlinear situations, SVMs utilize *kernel* methods, which enable the construction of linear classifiers in high-dimensional feature spaces that are nonlinearly related to input space [10].

Support Vector Classification

Chih-Chung and Chih-Jen give the following formalization of the Support Vector Classification (SVC) in [3]. Given training instances $\mathbf{x}_i \in \mathbb{R}^d, i = 1, \ldots, n$ in two classes, and a vector $\mathbf{y} \in \mathbb{R}^n$ of all outputs, such that $y_i \in \{1, -1\}$, SVC solves the following primal optimization problem:

$$\min_{\boldsymbol{\omega}, b, \boldsymbol{\zeta}} \qquad \frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\omega} + R\sum_{i=1}^{n}\zeta_i \tag{3.1}$$

subject to

$$\begin{cases} y_i(\boldsymbol{\omega}^T\phi(\mathbf{x}_i) + b) \geq 1 - \zeta_i \\ \zeta_i \geq 0, \qquad i = 1, \ldots, n, \end{cases} \tag{3.2}$$

where $\phi(\mathbf{x}_i)$ maps $\mathbf{x}_i$ into a higher-dimensional space and $R > 0$ is the *regularization parameter*. Due to the possible high dimensionality of the vector variable $\boldsymbol{\omega}$, usually the following dual problem is solved [3]. Let $\mathbf{e}$ be the vector of all ones and $R > 0$ the upper bound. The dual problem is then given by:

$$\min_{\boldsymbol{\alpha}} \qquad \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \tag{3.3}$$

subject to

$$\begin{cases} \mathbf{y}^T\boldsymbol{\alpha} = 0 \\ 0 \leq \alpha_i \leq R, \qquad i = 1, \ldots, n, \end{cases} \tag{3.4}$$

where $\mathbf{Q}$ is an $n \times n$ positive semidefinite matrix, $\mathbf{Q}_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, where $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ is the *kernel*. The training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function ϕ.

After the dual problem is solved, using the primal-dual relationship, the optimal $\boldsymbol{\omega}$ satisfies

$$\boldsymbol{\omega} = \sum_{i=1}^{n} y_i\alpha_i\phi_i(\mathbf{x}_i). \tag{3.5}$$

The decision function is then:

$$\operatorname{sgn}\left(\boldsymbol{\omega}^T\phi(\mathbf{x}) + b\right) = \operatorname{sgn}\left(\sum_{i=1}^{n} y_i\alpha_i K(\mathbf{x}_i, \mathbf{x}) + b\right). \tag{3.6}$$

The *kernel function* can be one of the following [24].

- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$,
- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma\langle \mathbf{x}_i, \mathbf{x}_j \rangle + r)^d \cdot d$, for degree d,
- RBF: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2)$,
- sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma\langle \mathbf{x}_i, \mathbf{x}_j \rangle + r)$,

- chi-squared: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \sum_k \frac{(\mathbf{x}_{i,k} - \mathbf{x}_{j,k})^2}{\mathbf{x}_{i,k} + \mathbf{x}_{j,k}}\right)$,
- manually defined custom kernel.

Note, that here the notation $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ is an inner product and not a tuple, different from Section 2.3.1.

Random Forest Classifier

Random forests use multiple learning algorithms and hence, are an ensemble learning method well suited for classification. Random Forests are actually a modification of bagging, in which the essential idea is to average many noisy but approximately unbiased models, and hence reduce variance [7]. Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest [1]. Using a random selection of features to split each node yields error rates relatively small, but are more robust with respect to noise [1]. This makes random forest classifiers well suited for this particular classification problem, as the classification performance is relatively fast together with tendency to relief the overfitting of traditional decision trees. The following definition for random forest is adapted from [1].

Definition 3.1.1. A Random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(\mathbf{x},\Theta_i), i = 1, \ldots, m\}$ where the $\{\Theta_i\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input $\mathbf{x}$. △

In this thesis, Scikit-learn library's implementation of Random Forest Classifier is applied. The main difference to the original definition in [1] is that the Scikit-learn implementation combines classifiers by averaging their probabilistic prediction, instead of letting each classifier vote for a single class [24]. The main parameters of Random Forest Classifier in Scikit-learn library's implementation are $n\_estimators$, $max\_features$, $min\_samples\_split$, and max_depth. The parameter $n\_estimators$ denotes the number of trees in the forest and $max\_features$ denotes size of the random subsets of features to consider when splitting a node [24]. The parameter $min\_samples\_split$ denotes the minimum number of samples required to split an internal node, and max_depth is the maximum depth of the tree [24].

3.1.2 Clustering algorithms

k-means

k-means is a traditional clustering algorithm, often referred to as Lloyd's algorithm [24]. The algorithm divides a set of n instances into disjoint clusters $G = \{G_1, \ldots, G_k\}$, each described by the mean μ_i of the instances in the cluster G_i. These means are often called the *cluster centroids*, even though they might not be actual instances of X. The algorithm aims to separate the instances in k groups of equal variance, and uses *inertia* as the minimization criteria. Inertia can be seen as the measure of how internally coherent clusters are:

$$\arg\min_{G} \sum_{i=1}^{k} \sum_{\mathbf{x} \in G_i} \|\mathbf{x} - \mu_i\|^2 . \tag{3.7}$$

k-means scales well to large number of instances and is computationally fast. This makes it a good candidate in the context of reflection data. The drawbacks of the k-means algorithm are that it requires the number of clusters to be specified before the computation and hence, is sensitive to the initial choice of the number of clusters and the cluster centroids. This issue is addressed in Scikit-learn's algorithm by implementing k-means++ initialization scheme, which initializes the cluster centroids to be distant from each other [24]. k-means always converges, but may be to the local minimum. It makes the assumption of convex clusters, which is not necessarily the case with reflection data. It can also perform poorly with clusters of irregular shapes. In this thesis, k-means is used as a comparison algorithm, since several evaluation scores can be computed and compared against the number of clusters, and against results provided by other algorithms.

Affinity Propagation

Affinity Propagation (AP) is a clustering method based on passing messages between data points, introduced by Frey and Dueck in 2007 [6]. Unlike many traditional clustering algorithms, AP does not require determining the number of clusters before running the algorithm, but rather simultaneously considers all data points as potential exemplars [6]. While the traditional algorithms like k-means work well if the initial choice of the exemplars is close to a good solution [6], they do not apply so well if the number of clusters is unkown or a subject of examination, as the solutions are sensitive to the initial choice of exemplars.

Frey and Dueck explain in [6], that by viewing each data point as a node in a network, AP recursively transmits messages along edges of the network until a good set of exemplars and corresponding clusters is obtained. The algorithm is described in [6] as follows. AP takes as input a collection of real-valued *similarities* $sim(i, j)$

between data points, as well as the *preferences* $sim(i,i)$ for each data point i. The similarity $sim(i,j)$ indicates how well the data point with index j is suited to be the exemplar for data point i. The similarity is set by default to be the negative squared error (euclidean distance), but the method can be applied with different optimization criteria set by hand as well. Data points with larger preferences are more likely to be chosen as exemplars. This value can be varied to produce different number of clusters, but it should be set to a common value in case all the data points are equally suitable as exemplars. There are two kinds of messages exchanged between data points: *responsibility* $res(i,j)$ and *availability* $ava(i,j)$. Responsibility represents the accumulated evidence for how well-suited the point j is to serve as the exemplar for point i. Availability, on the other hand, represents the accumulated evidence for how appropriate it would be for a point i to choose point j as its exemplar. Responsibility and availability can be viewed as log-probability ratios.

At the beginning, the availabilities are initialized to zero and the responsibilities are calculated in the following manner:

$$res(i,j) \longleftarrow sim(i,j) - \max_{j',j' \neq j} \{ava(i,j') + sim(i,j')\}. \tag{3.8}$$

The availability update is calculated as follows:

$$ava(i,j) \longleftarrow \min\{0, res(j,j) + \sum_{i',i' \notin \{i,j\}} \max\{0, res(i',j)\}\}. \tag{3.9}$$

The "self-availability", $ava(j,j)$, is updated differently:

$$ava(j,j) \longleftarrow \sum_{i',i' \neq j} \max\{0, res(i',j)\}. \tag{3.10}$$

At any point during the run, availabilities and responsibilities can be combined to identify exemplars: For point i, the value j that maximizes $ava(i,j) + res(i,j)$ either identifies point i as an exemplar if $j = i$, or identifies the data point that is exemplar for point i. The message-passing procedure may, hence, be terminated after fixed number of iterations, or after the local decisions stay constant for some number of iterations. In this thesis, Affinity Propagation is implemented with Scikit-learn library [24].

HDBSCAN

Density-based clustering methods like DBSCAN are popular, but have number of limitations: Some methods provide only non-hierarchical labeling using a global density threshold. This is often a problem in situations with clusters of different densities and nested clusters [2]. On the other hand, among those methods that sup-

port clustering hierarchy, common limitation is the representation of the hierarchy, or that the method only supports specific kinds of problems [2].

HDBSCAN is a clustering method extending previously developed DBSCAN algorithm. According to the authors Campello et al., HDBSCAN is both theoretically and practically improved, density based, hierarchical method outperforming the other current density-based methods on a wide variety of real world data [2]. The significant feature of HBDSCAN is that, according to the authors, it is not suffering from any of the limitations mentioned above [2]. Similar to Affinity Propagation, HDBSCAN does not require the number of clusters to be estimated beforehand. These qualities make HDBSCAN another appealing candidate for clustering tasks on the reflection data.

Campello et al. give the following optimization algorithm for HDBSCAN in [2]. Let $\{G_2, \ldots, G_k\}$ be the collection of all clusters in the simplified cluster hierarchy generated by HDBSCAN, except the root G_1, and let $S(G_i)$ denote the stability value of each cluster. Then, the task of maximizing the sum of stabilities of the extracted clusters is the optimization problem:

$$\max_{\xi_2,\ldots,\xi_k} \sum_{i=2}^{k} \xi_i S(G_i) \tag{3.11}$$

subject to

$$\begin{cases} \xi_i \in \{0,1\}, & i = 2, \ldots, k \\ \sum_{j \in I_h} \xi_j = 1, \forall h \in L, \end{cases} \tag{3.12}$$

where $\xi_i, (i = 2, \ldots, k)$, indicates whether cluster G_i is included in the flat solution or not. $L = \{h \mid G_h \text{ is a leaf cluster}\}$ is the set of indices of leaf clusters, and $I_h = \{j \mid j \neq 1 \text{ and } G_j \text{ is ascendant of } G_h\}$ is the set of indices of all clusters on the path form G_h to the root. In this thesis, the codebase provided by McInnes et al. in [19] is used for applying HDBSCAN.

3.2 Creating and combining features

3.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a linear method for dimensionality reduction for multivariate data, developed by Pearson [23] and Hotelling [8]. PCA transforms the data to a new coordinate system, such that the new variables, *the principal components*, are uncorrelated, linear combinations of the original variables. The objective is thus, to reduce the number of components while preserving the maximal amount of relevant information. In addition to dimensionality reduction, PCA can be used to discover important features of the data, such as outliers and distributional

characteristics, based on the principal component scores [10]. The most common uses of the method are, however, data preprocessing, visualization, image analysis, and pattern recognition.

In the context of reflection data, PCA provides means for projecting the data into a lower dimensional space. This, in turn, provides the means for visualization, suitable for human observation. The drawback in using PCA is that being a linear algorithm, complex relationships between high-dimensional features may not be interpreted correctly and some information might be lost in the process. To address this issue, a more sophisticated method, t-SNE, is introduced in the next section.

The visualizations obtained with PCA are used in classification and clustering tasks to gain more information about the process and results. Probabilistic PCA model as implemented in this work was published by Tipping and Bishop in [33]. They formulate PCA within maximum likelihood framework and summarize the original definition by Hotelling [8] as follows:

Definition 3.2.1. For a set of observed d-dimensional data vectors $\{\mathbf{x}_i\}$, $i = 1, \ldots, n$, the q principal axes, $\mathbf{w}_j$, $j = 1, \ldots, q$ are those orthonormal axes onto which the retained variance under projection is maximal. △

Here the vectors $\mathbf{w}_j$ are given by q dominant eigenvectors of the sample covariance matrix $\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$ such that $\mathbf{S}\mathbf{w}_j = \lambda_j \mathbf{w}_j$, and $\bar{\mathbf{x}}$ is the sample mean. The vector $\mathbf{t}_i = \mathbf{W}^T(\mathbf{x}_i - \bar{\mathbf{x}})$, where $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_q)$ is a q-dimensional reduced representation of the observed vector $\mathbf{x}_i$.

3.2.2 t-SNE

For nonlinear dimensionality reduction, a more recent and advanced algorithm called t-Distributed Stochastic Neighbor Embedding (t-SNE) was developed in 2008 by van der Maaten [34]. It is an improved version of Stochastic Neighbor Embedding (SNE) algorithm. Like PCA, t-SNE can also be used to explore high-dimensional data in two or more dimensions. It is based on probability distributions focused on retaining both local and global structure of the data.

The main difference between SNE and the improved version, t-SNE, is the cost function [34]. The original cost function, which aims to minimize the sum on Kullback-Leibler divergences over all data points using gradient descent method, is replaced by a symmetric version with simpler gradients. t-SNE also uses Student-t distribution to compute similarities between points in the low-dimensional space, which makes it easier to optimize the cost function [34]. The symmetric SNE aims to minimize a single Kullback-Leibler divergence between a joint probability distribution, P, in the higher dimensional space and a joint probability distribution, Q,

in the lower-dimensional space.

$$Cost_{\text{t-SNE}} : KL\left(P \mid Q\right) = \sum_{i} \sum_{j} p_{ij} \log_2 \frac{p_{ij}}{q_{ij}}. \tag{3.13}$$

Symmetry comes from the property that $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}$, $\forall i,j$. Here, the pairwise similarities in the low-dimensional space are given by:

$$q_{ij} = \frac{\exp\left(-\left\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\right\|^2\right)}{\sum_{k \neq l} \exp\left(-\left\|\tilde{\mathbf{x}}_k - \tilde{\mathbf{x}}_l\right\|^2\right)}, \tag{3.14}$$

and the pairwise similarities in the high-dimensional space are given by:

$$p_{ij} = \frac{\exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 / 2\sigma^2\right)}{\sum_{k \neq l} \exp\left(-\left\|\mathbf{x}_k - \mathbf{x}_l\right\|^2 / 2\sigma^2\right)}, \tag{3.15}$$

where $\tilde{\mathbf{x}}_i$ are low-dimensional data points mapped from high-dimensional data points $\mathbf{x}_i$ and σ is the variance.

3.2.3 Point Feature Histograms

Point Feature Histogram (PFH) is a statistical representation of three-dimensional shapes, presented by Wahl et al. in [36] and later extended by Rusu et al. in [30]. The PHFs presented in [36] are based on a four-dimensional feature, which aims to parameterize the geometrical relation of an *oriented surface-point pair*. In [36], the set of such features represents both local and global characteristics of the surface, represented by a single histogram, which is then applied to 3D object recognition. In [30] this idea is extended by computing local point feature histograms for each point in the cloud to characterize the surface on which the points lie. In this thesis, the Point Cloud Library's [29] implementation of PFHs based on [28], is applied to the reflection point cloud data in order to create new features for classification.

Normal estimation

In order to compute the PFH descriptors for the point cloud data, surface normals need to be estimated for each point in the point cloud. In the Point Cloud Library's implementation of normal estimation, the problem of determining the normal to a point on the surface is approximated by the normal estimation of a plane tangent, which then becomes a least-square plane fitting estimation problem [28]. Therefore the problem is reduced to an analysis of the eigenvectors and eigenvalues of a covariance matrix created from the nearest neighbors of the query point. Formally, for each point p_i, the covariance matrix $\mathbf{S}$ is defined as follows [28]:

$$\mathbf{S} = \frac{1}{k}\sum_{i=1}^{k}(p_i - \bar{p}) \cdot (p_i - \bar{p})^T, \qquad \boldsymbol{S} \cdot \mathbf{v_j} = \lambda_j \cdot \mathbf{v}_j, \qquad j \in \{0,1,2\}, \tag{3.16}$$

where k is the number of neighbors considered in the neighborhood of p_i, $\bar{p}$ is the centroid of the nearest neighbors, λ_j is the j-th eigenvalue of the covariance matrix, and $\mathbf{v}_j$ the corresponding eigenvector. Note that in the context of PFHs, k is again different from the usual (number of clusters) because of the established notation k-neighborhood.

Since the sign of the normal can not be formally solved, the viewpoint is by default set to origin, and can be re-orinented manually. As surface normals need to be estimated based on the support of surrounding point neighborhood (k-neighborhood), the threshold used in nearest-neighbor estimation is an important factor of normal estimation [28]. More specifically, In Point Cloud Library's implementation, the choice of treshold (either radius r of neighboring point search or the number k of closest neighbors in neighboring point search) becomes an important issue. If the treshold chosen is too large, the set of neighbors may cover points from adjacent surfaces, which is not desired situation, since the resulting PFH descriptors may get distorted. On the other hand, if the treshold chosen is too small, there might not be enought neighboring points to support the normal estimation, therefore resulting in infinite normals. Thus, the right scale needs to be chosen based on the level of detail of the application in question [28]. In the context of reflection data, the Point Cloud Library's implementation of normal estimation is not used, but the normals are computed manually for each point, using the information of *structured* point clouds and neighboring points. This approach is chosen because of the sparseness of the point clouds in reflection data. In order to find enough neighboring points for normal estimation, the radius of neighbor-search becomes inevitably too large containing almost the entire defect, and hence, the normals become distorted.

PFH descriptors

The general idea of PFH descriptors is to represent a point's k-neighborhood geometrical properties with histogram of four-dimensional features, based on the relationships of the points in the k-neighborhood and their surface normals [28]. More specifically, the resulting PFH descriptor is a histogram of relationships between all pairs of points in the neighborhood. Hence, the PFHs are highly dependent on the quality of surface normal estimation [28]. The mathematical formulation, adapted from [28] the following:
A fixed coordinate frame is defined to compute the relative difference between points p_i and p_j:

$$\begin{cases} \mathbf{u} = \mathbf{n}_s \\ \mathbf{v} = \mathbf{u} \times \dfrac{(p_t - p_s)}{\|p_t - p_s\|_2} \\ \mathbf{w} = \mathbf{u} \times \mathbf{v} \end{cases} \tag{3.17}$$

Then, using the above **uvw** frame, the difference between normals $\mathbf{n}_s$ and $\mathbf{n}_t$ can be expressed as a set of angular features:

$$\begin{cases} \alpha = \mathbf{v} \cdot \mathbf{n}_t \\ \phi = \mathbf{u} \cdot \dfrac{(p_t - p_s)}{\delta_E} \\ \theta = \text{atan2}(\mathbf{w} \cdot \mathbf{n}_t, \mathbf{u} \cdot \mathbf{n}_t) \end{cases} \tag{3.18}$$

where δ_E is the euclidean distance between the two points p_s and p_t, and atan2 is the two argument variant of arctangent:

$$\text{atan2}(\mathbf{w} \cdot \mathbf{n}_t, \mathbf{u} \cdot \mathbf{n}_t) = \arctan\left(\frac{\mathbf{w} \cdot \mathbf{n}_t}{\mathbf{u} \cdot \mathbf{n}_t}\right).$$

The quadruplet $\langle \alpha, \phi, \theta, \delta_E \rangle$ is computed for each pair of two points in k-neighborhood. The set of all quadruplets is then binned into a histogram, which constructs the final PFH representation for the query point. However, in some cases the feature δ_E has proven to be insignificant [28]. By default, the Point Cloud Library implementation uses 5 binning subdivisions and does not include the feature δ_E.

3.3 Empirical Evaluation

3.3.1 Research Questions

The key questions to which this thesis aims to answer based on empirical evaluation are the following:

1. Whether internal structure exists in the data set, and if so, whether it has something in common with the known class labeling.
2. Whether distinct learning algorithms perform similarly on the classification task.
3. Whether Point Feature Histograms are useful in creating new features and whether these features can improve the classification and clustering results.

In order to answer these questions, empirical evaluation is divided into classification and clustering tasks. The results are then evaluated for several different algorithms introduced in Section 3.1. Finally, new features are added into the original data set and clustering and classification results are then evaluated again.

3.3.2 Description of the Data

The data set used in this thesis is referred to as *reflection data*. It is obtained by using Phase Measuring Deflectormetry, thus reflecting and phase shifting sine patterns on the inspected surface in two different directions. The reflection is then observed by four cameras. Any defects on the surface will cause deviations in the pattern, which are then evaluated and detected by a software. Reflectivity, brightness and local curvature of the detected object are then calculated out of these raw phase images. An example of the same defect showing on the inspected surface on amplitude (reflectivity) and curvature channels can be seen in Figure 3.1.

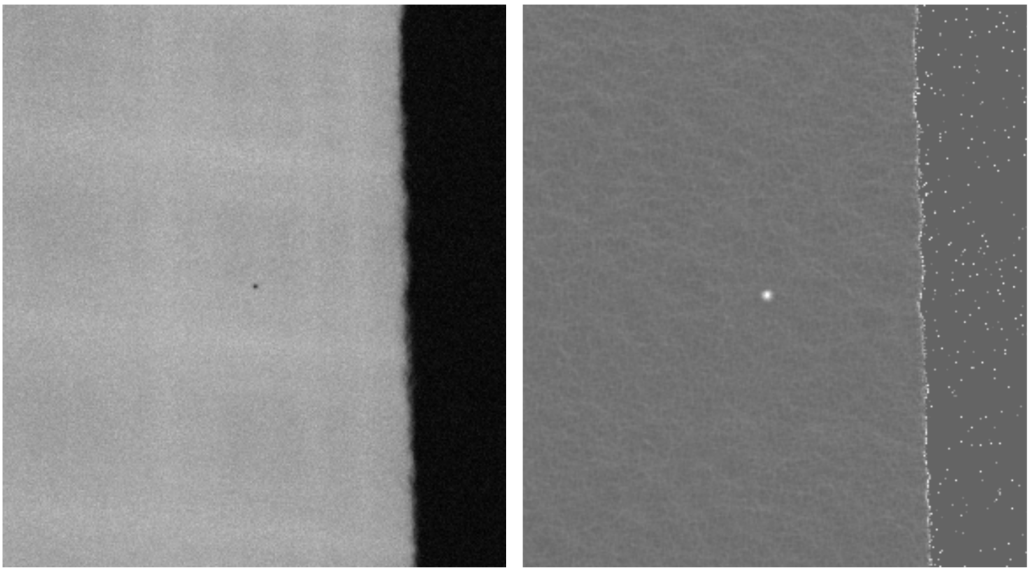

Figure 3.1: The same defect showing on amplitude (reflectivity) channel on the left and curvature channel on the right.

The reflection data consists of two separate parts. First part is an excel-table of $n = 4627$ examples with 10 distinct categorical defect classes, classified by human inspectors. Eight of the classes are real defect categories whereas class 9 is a pseudo-class. Class 10 is a class of instances that could not be assigned to any other category, essentially consisting of unlabeled instances. Therefore, the set of possible output values is

$$Y = \{\text{class1}, \text{class2}, \text{class3}, \text{class4}, \text{class5}, \text{class6}, \text{class7}, \text{class8}, \text{class9}, \text{class10}\}.$$

The data can be characterized as unbalanced, since class 1 examples cover approximately 81.5% of the entire set of examples, whereas only one example of both class 8 and class 3 exist in the data. This unbalance poses the biggest challenge related to the learning tasks in this thesis.

The features are generated from *Binary Large Object* (BLOB) analysis, and 3D reconstruction of the defect. BLOB refers to a group of connected pixels in a binary image, and the features computed based on BLOB-analysis aim to characterize the defect shape and size [21]. One important feature of a BLOB is a bounding box, which is the minimum rectangle containing the BLOB [21]. In the context of reflection data, features such as radius of the BLOB, rectangularity of the BLOB, perimeter of the BLOB, and pixel-size of the BLOB are included. More detailed descriptions about the common BLOB features and the extraction process can be found in [21]. The rest of the features are extracted from the 3D data of the defect. For example, the height or depth of the defect, or a ratio of two features like lw-ratio (length/width) and hd-ratio (height/diameter) are included.

The second part of reflection data consists of point clouds in csv-format. The point cloud is a representation of the shape of the defect. Surface plots of point clouds related to a single instance of each class can be seen in the Figure 3.2. Each point cloud corresponds to a single example in the first part of the data. More comprehensive description of reflection data is property of Micro-Epsilon and hence, it is not given here. However, all the feature names are given in the next section.

Feature Selection

There are 38 features originally. However, some of the features are strictly related to the detection and reconstruction process of surface defects, and are considered to be insignificant in the context of learning tasks. Therefore, before applying learning algorithms, only the important ones are selected from these features. The resulting set of 20 features, referred to as the *original features*, are listed in Table 3.1. The first feature listed in the Table 3.1, called "kernel-channel" is a categorical feature referring to the channel in which the defect was detected. This might be useful information for learning tasks and therefore, kernel-channel was included in the set. All other features are numerical.

In addition to the original 20 features, a gaussian function was previously fitted to the point cloud data, resulting in 13 features. These features are listed in Table 3.2 and more detailed description about the generation process and formulation can be found in Appendix B. Finally, additional PFH descriptors were computed, resulting in 125 new features. The generation process is explained in Section 3.3.5. All of these features are not used in classification and clustering tasks, but feature importance is used to determine the most important features regarding the learning tasks in Section 4.3.1.

Adding all together, there are three distinct sets of features used in this thesis: The*original features*, the *gaussian features*, and the *PFH-features*, recording values of different attributes of the same 4627 examples. To clarify which data set is in

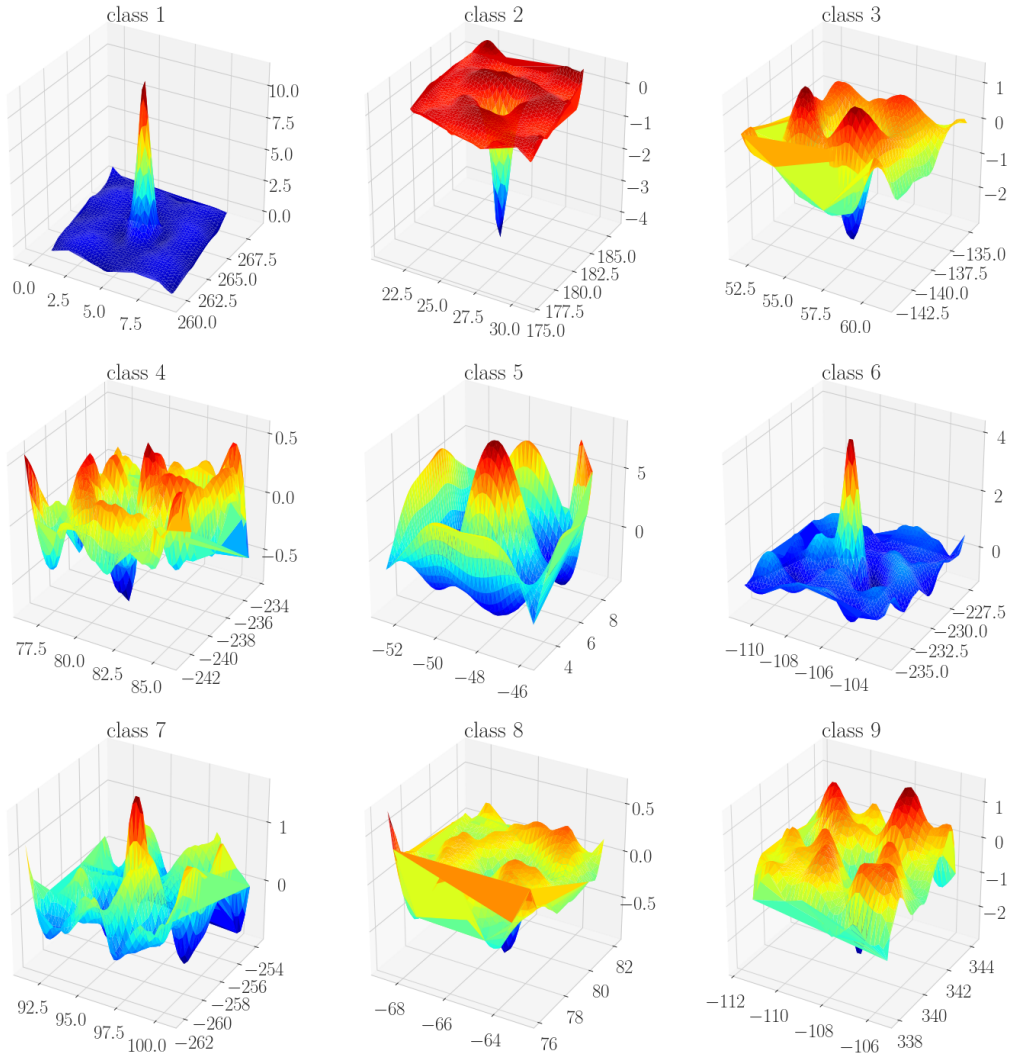

Figure 3.2: 3D-surface plots of point cloud examples of each class, except class 10, which is the class of unlabeled examples.

question when presenting the results, let us call these sets in the following way.

- F_{og} = original features
- F_{gauss} = gaussian features
- F_{PFH} = PFH-features

Classification is first applied to the data set containing the original and gaussian features. Let us call this set of features

$$F1 = F_{og} \cup F_{gauss}.$$

Clustering is then applied to the data set containing only the original features, F_{og}. Classification and clustering is also applied to the data set containing the best

combination of all of these features, including features based on PFH descriptors, in order to compare the results. This set of features is called F2 and explained in more detail in Section 4.3.1.

Table 3.1: The original features, called F_{og}

1. kernel-channel
2. blob-area
3. blob-perimeter
4. blob-formfactor
5. blob-rectangularity
6. blob-avgradius
7. blob-minradius
8. blob-maxradius
9. principal axis-distance variance major axis
10. principal axis-distance variance minor axis
11. principal axis-major axis u
12. principal axis-major axis v
13. reconstruction-zmin
14. reconstruction-zmax
15. reconstruction-rms
16. lw-ratio
17. defect-z
18. hd-ratio
19. z_min / z_rms
20. z_max / z_rms

Table 3.2: Features computed based on fitting gaussian functions on point cloud data. F_{gauss}

1. gauss-amplitude z
2. gauss-offset z
3. gauss-center x
4. gauss-center y
5. gauss-sigma major axis
6. gauss-sigma minor axis
7. gauss-orientation angle
8. gauss-fitting error
9. gauss-fitting rms
10. gauss-volume
11. volume positive
12. volume negative
13. volume total

Min-max Scaling

Several machine learning methods, including SVM, assume that the features in the data set are centered around zero and have variance of same magnitude [24]. This is because features with larger magnitudes of variance might dominate the objective function and hence, affect the learning results. It is a common practice to scale the data either to zero mean and unit variance (*standardization*), or scale all variables in the range [0,1] (*normalization*). In the context of reflection data, the features are normalized together in the preprocessing stage, before applying the learning algorithms.

3.3.3 Classification methods

Classification is applied to the data set using machine learning library from Python, called *Scikit-learn*. Most classification algorithms are designed for binary classification, and in the context of reflection data, it is beneficial to perform *binary classification*, since the data set is heavily unbalanced. This also solves the issue of some of the classes containing only one or two examples. It is a matter of interest to be able to separate the biggest and most common category, *class 1*, from other classes. Hence, the original categorical labeling is first converted to binary numerical labeling by replacing all the class 1 labels with value 1, and any other class labels with value 0 ("others"). Thus, $Y_{\text{binary}} = \{0, 1\}$.

In addition to binary classification, *multiclass classification* is performed in order to distinguish the four biggest classes; 1, 2, 4, and 6 from the others. This is done by changing the labeling such that classes 1, 2, 4, and 6 are left with their original numerical labels but all other classes are labeled as 0 ("others"). This approach yields to total of 5 separate classes, hence $Y_{\text{multi}} = \{0, 1, 2, 4, 6\}$. Multiclass classification is also further experimented on *balanced data set* because of the heavy unbalance of the different classes. This means, that an equal number of samples of each class is included in the training set in order to generate such set that none of the classes is dominating over others. This approach is taken in order to see whether it is actually possible to distinguish between all the classes containing at least 22 samples. Such classes are class 1, class 2, class 4, class 6, class 7, and class 9. Class 10 is again dropped out from the classification as well as classes 3, 5, and 8 that only contain one or two examples each. Thus, $Y_{\text{balanced}} = \{1, 2, 4, 6, 7, 9\}$.

Next, models are generated using two different algorithms, Random Forest Classifier and SVC (introduced in Section 3.1). The performance of these models are evaluated using accuracy, cross-validation score, and confusion matrix. Models are also created from such data set, that all the class 10 samples are left out. In multiclass cases, class 10 was always dropped out. Class 10 samples are essentially

unlabeled, so it is not clear whether it is beneficial to use them in training and testing of models. Finally, different models are compared in order to answer the research question regarding the performance of different algorithms.

3.3.4 Clustering methods

Before applying clustering algorithms to the data, it is important to evaluate, whether the data actually has some internal structure and clustering tendency or not. In the context of reflection data, direct visual assessment of clustering tendency is not possible, since the data set is multidimensional. However, it is possible to transform the data with PCA or with t-SNE into a lower dimensional projection. This makes it possible to assess the internal structure in a visual manner. Even though some information might be lost in the dimensionality reduction process, the overall structure and the most significant components still remain, and more insight about the structure of data can be obtained.

Clustering is applied to the data set using three different algorithms: k-means, Affinity Propagation, and HDBSCAN. In both internal and external evaluation, distinct sets of clusters obtained with different algorithms are compared. First, silhouette analysis is used to determine such number of clusters that the resulting clustering will fit the data without any external information about the correct class labels. Higher silhouette scores imply that the clustering fits the internal structure of the data, the distances between instances belonging to same cluster being relatively small compared to the distances between instances of separate clusters, whereas lower silhouette scores imply that the clustering does not fit the data. Higher silhouette scores indicate well separated clusters and hence, internal structure being present, whereas lower silhouette scores indicate overlapping or random clusters. This information is also used in order to determine whether internal structure exists in the data.

Next, the obtained cluster labels are compared with correct class labels in order to explore whether there is something in common with the internal structure of the data and human-assigned class labels (ground truth labeling). External evaluation metrics, such as homogeneity, completeness, ARI, and AMI, are applied. The similarity between cluster labeling and ground truth labeling is also evaluated using PCA and t-SNE transformations, by visualizing the different clusterings and comparing the cluster labels to the ground truth labeling.

3.3.5 PFH methods

Computing PFH:s for a point cloud is a relatively straightforward process, since there are readily available packages in the Point Cloud Library [29], as well as examples of

correct usage. More challenging is to choose the correct radius for computations, so that very little information is lost, simultaneously keeping the radius large enough that there are enough neighboring points available to perform the computations. In this case, radius was not used for normal evaluation, but the point normals were computed manually, using the structure-information of point clouds. Usually some down-sampling is done before the computations, but in this particular situation it was not necessary, since the point clouds are already relatively sparse. Only the points inside the bounding box, in other words, points near to the actual defect were used for PFH-computations. This also makes it unnecessary to perform down-sampling. A visualization of a point cloud together with the related point normals can be seen from Figure 3.3.

Each resulting histogram consists of 125 bins (125 values) and there are as many histograms generated as there are points in the point cloud. An example of the resulting histograms of a single sample, plotted in a single figure, can be seen from Figure 3.4. In most implementations, histograms are compared and matched using some kind measure of similarity. In this thesis, however, the focus is on creating new features for classification and clustering tasks. Hence, the histograms need to be somehow combined to get a reasonable number of new features for a single sample. Next, cumulative sums are computed for each bin using all the histograms, in order to achieve a single histogram for every example in the original data. This process produces 125 new features per example. Cumulative sum histograms of examples belonging to classes 1, 2, and 5 together with related point clouds can be seen from Figures 3.5, 3.6, and 3.7. The Figures indicate, that even the cumulative histograms manage to capture some characteristics of the surface shape. For example, the size of the defect between Figures 3.5 and 3.6 is visible in the cumulative histogram. However, the histograms cannot distinguish between the directions (above or below the surface) of the defect shape.

Using feature importance on the cumulative histograms and other features for further dimensionality reduction, a combination data set from all the features is generated. Finally, this new data set, consisting of 20 features together with the output variable, is used in the classification and clustering tasks. More detailed description of this combination set of features follows in Section 4.3.1. The objective is to compare the new results with the ones obtained with original data set and to evaluate whether the PFH:s are useful in the context of reflection data and the learning tasks.

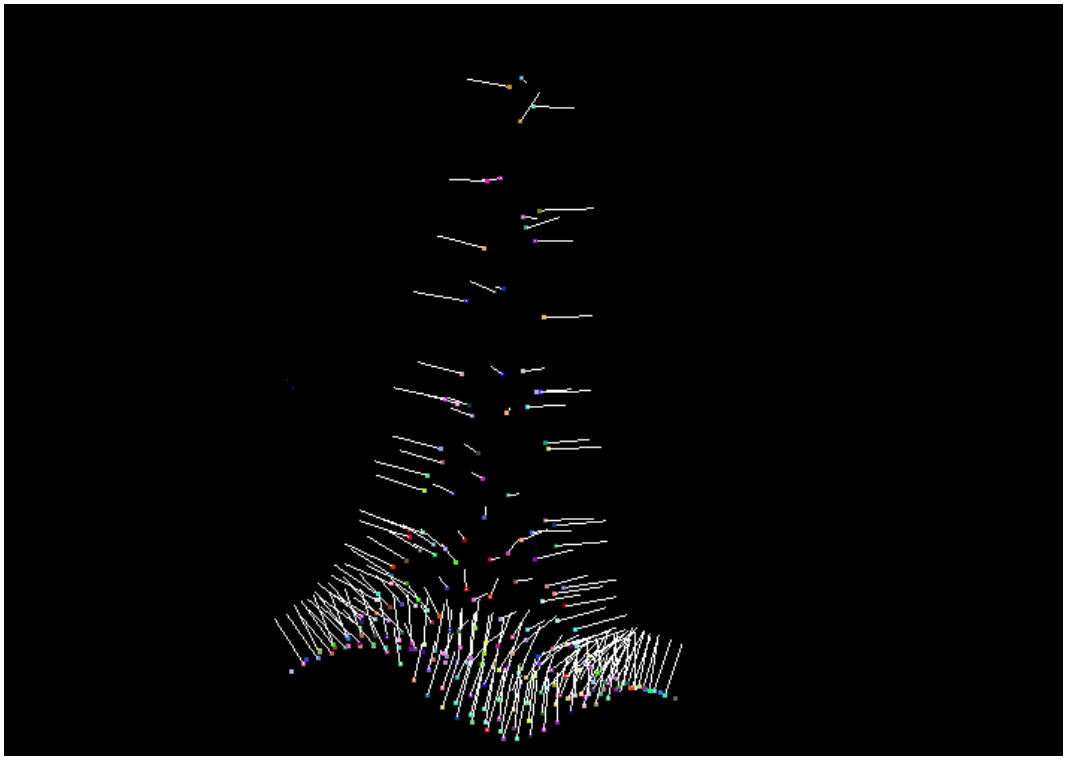

Figure 3.3: Point cloud and estimated point normals visualized using Point Cloud Libarary's visualizer. The points in the picture are only the points inside the bounding box. In other words, no surrounding points are visualized but only the points near the defect. The estimated normals are used in computing the PFH descriptors for each point.

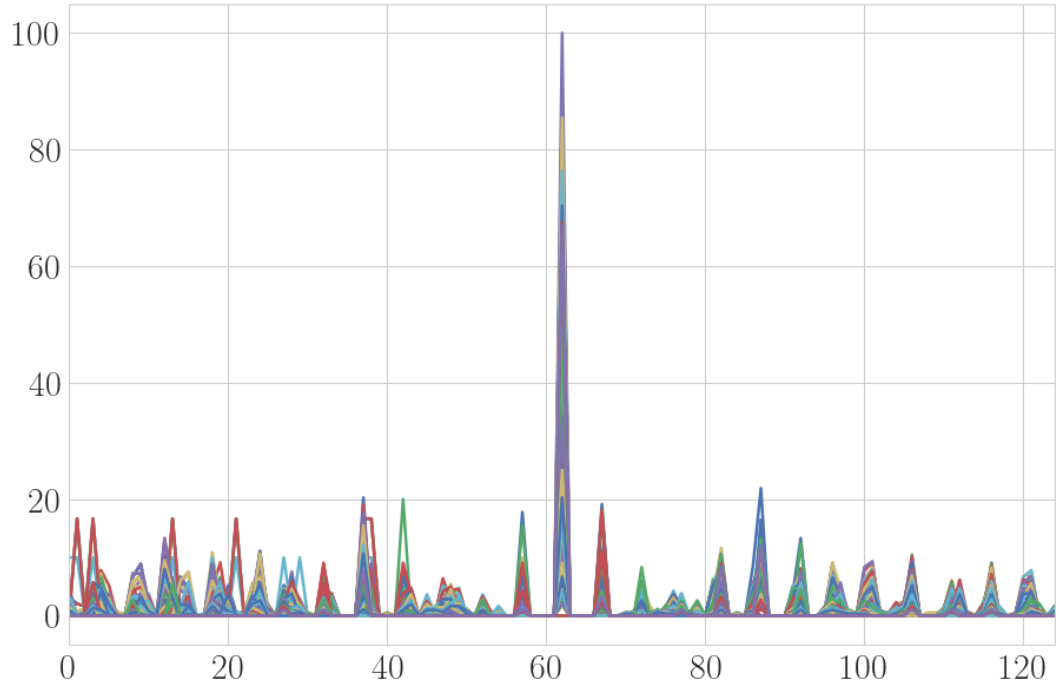

Figure 3.4: An Example of all the PFH descriptors related to a single point cloud of a single instance, plotted in the same figure.

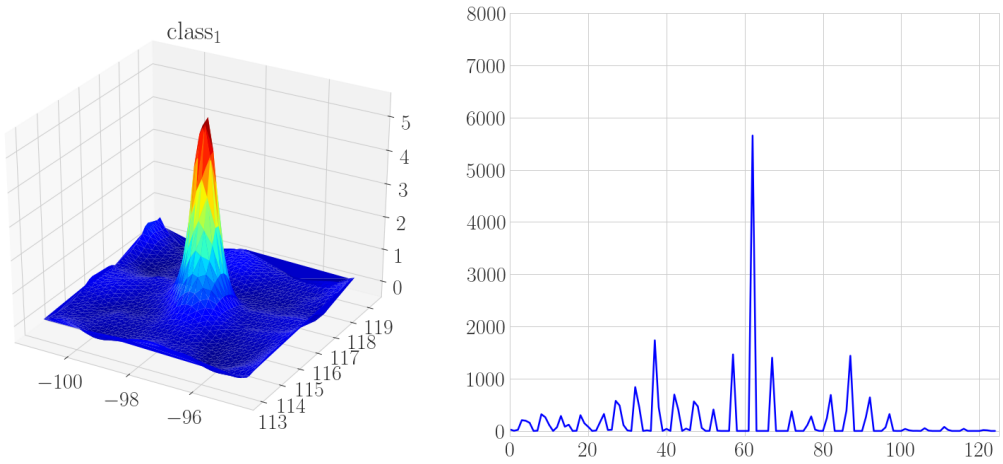

Figure 3.5: Cumulative sum histogram of class 1 example and related 3D point cloud plot.

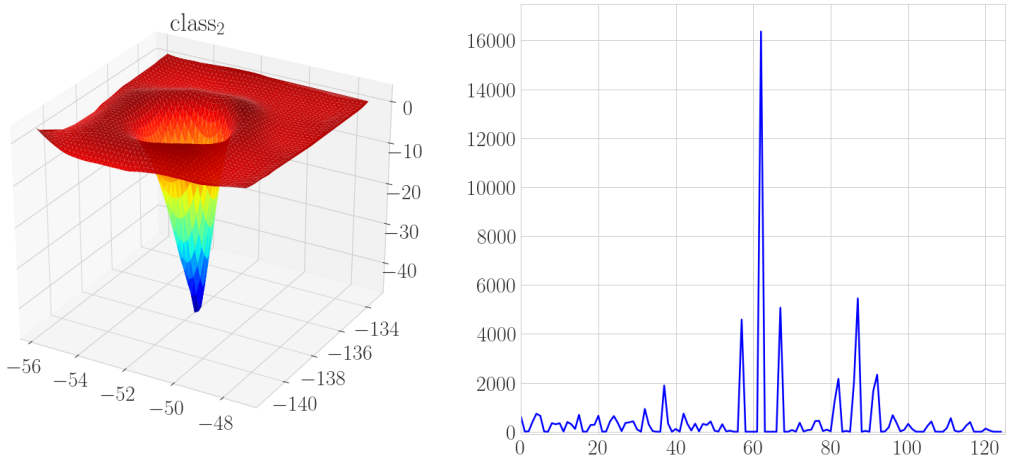

Figure 3.6: Cumulative sum histogram of class 2 example and related 3D point cloud plot.

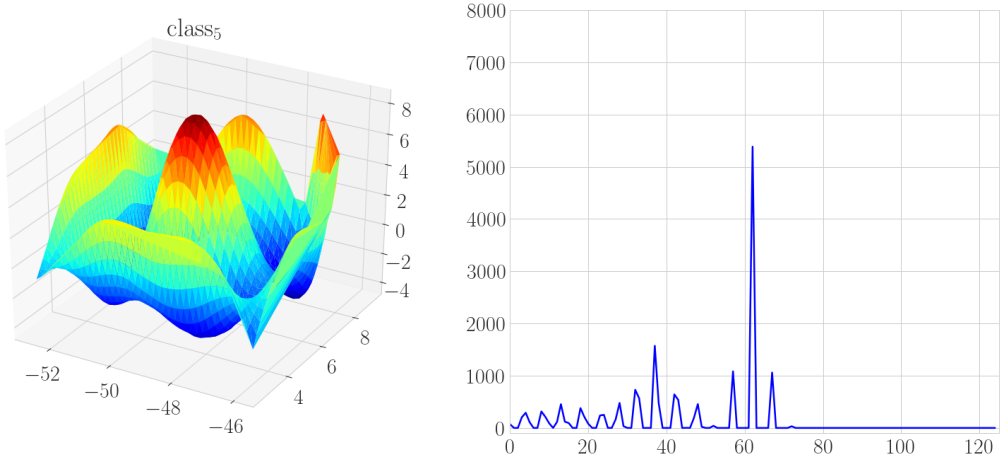

Figure 3.7: Cumulative sum histogram of class 5 example and related 3D point cloud plot.

4. RESULTS AND DISCUSSION

In this chapter, results of empirical evaluation are presented. The results are divided into classification and clustering tasks, and in more detail, into results obtained with different algorithms. First, classification results on the data set F1 obtained with the two different algorithms are presented. Next, clustering results on the data set F_{og} using three different algorithms are presented. Furthermore, classification and clustering results using features obtained by computing Point Feature Histograms (data set F2) are presented and compared with the previous results. Finally, some discussion about the different results as well as answers to some of the research questions are given.

4.1 Classification results

4.1.1 Random Forest Classifier results

For Random Forest Classifier, introduced in Chapter 3.1, the parameters chosen determine four limitations for the hypotheses considered by the algorithm. These limitations were chosen, since the algorithm is required to stay limited sized and relatively fast for large data sets. Thus, the hypothesis class becomes

$$\mathcal{H}_{\mathrm{RF}} = \{h \mid n_{estimators} = 30, min\_samples\_spilt = 10, max\_depth = 10, \\ max\_features = 1\}. \tag{4.1}$$

Random Forest Classifier provides binary classification results within $\mathcal{H}_{\mathrm{RF}}$ on the data set F1 with accuracy of **92.695**% and cross-validation score of **86.882**%, using 5 folds for the entire data set. From the Confusion Matrix (4.2), it can be seen that the model confuses as many as 276 examples to be class 1, which actually are class 0. On the other hand, the model confuses only 62 examples to be class 0, actually being class 1. This seems reasonable, since the data set is not balanced; approximately 81.5% of the examples belong to class 1 after replacing the original class labels with binary labels.

$$\mathbf{C}_{\mathrm{RF1}} = \begin{bmatrix} & \text{class0} & \text{class1} \\ \text{class0} & 570 & 276 \\ \text{class1} & 62 & 3719 \end{bmatrix} \tag{4.2}$$

When dropping all class 10 (unlabeled) examples from the data set and repeating the process on data set F1, Random Forest Classifier provides binary classification results with accuracy of **96.780**% and cross-validation score of **91.770**%. The results improve significantly when class 10 is left out from the data. The confusion matrix without class 10 is the following:

$$\mathbf{C}_{\mathrm{RF2}} = \begin{bmatrix} & \text{class0} & \text{class1} \\ \text{class0} & 284 & 128 \\ \text{class1} & 7 & 3774 \end{bmatrix} \tag{4.3}$$

Finally, when applying Random Forest Classifier to the multiclass situation on the set F1, and trying to distinguish classes 1, 2, 4, and 6 from others, the following results are obtained: Accuracy of **95.969**% and cross-validation score of **94.086**%. From the Confusion Matrix (4.4), it can be seen that while accuracy is relatively high, the model classifies correctly only class 0, 1, and 2 examples, but confuses class 4 and 6 examples as class 1 examples. As many as 59 examples belonging to class 0 are also classified as class 1. High accuracy in multiclass classification therefore follows from the large proportion of class 1 examples, which are correctly classified.

$$\mathbf{C}_{\mathrm{RF3}} = \begin{bmatrix} & \text{class0} & \text{class1} & \text{class2} & \text{class4} & \text{class6} \\ \text{class0} & 146 & 59 & 0 & 0 & 0 \\ \text{class1} & 5 & 3773 & 2 & 0 & 0 \\ \text{class2} & 1 & 16 & 102 & 0 & 0 \\ \text{class4} & 0 & 53 & 0 & 3 & 0 \\ \text{class6} & 2 & 30 & 1 & 0 & 0 \end{bmatrix} \tag{4.4}$$

The results obtained with Random Forest Classifier on the set F1 are summarized in Table 4.1.

Table 4.1: Results obtained with Random Forest Classifier and features F1

	All classes included	Without class 10	Multiclass case
Accuracy	92.695 %	96.780 %	95.969 %
Cross-Val Score	86.882 %	91.770 %	94.086 %

4.1.2 Support Vector Classifier results

Support Vector Classifier makes it possible to specify the kernel type used in the algorithm, default being RBF in Scikit-learn library's implementation. The results obtained with different kernels and using data set F1 can be seen from Figure 4.1. The best choice for kernel in the context of reflection data seems to be chi-squared kernel.

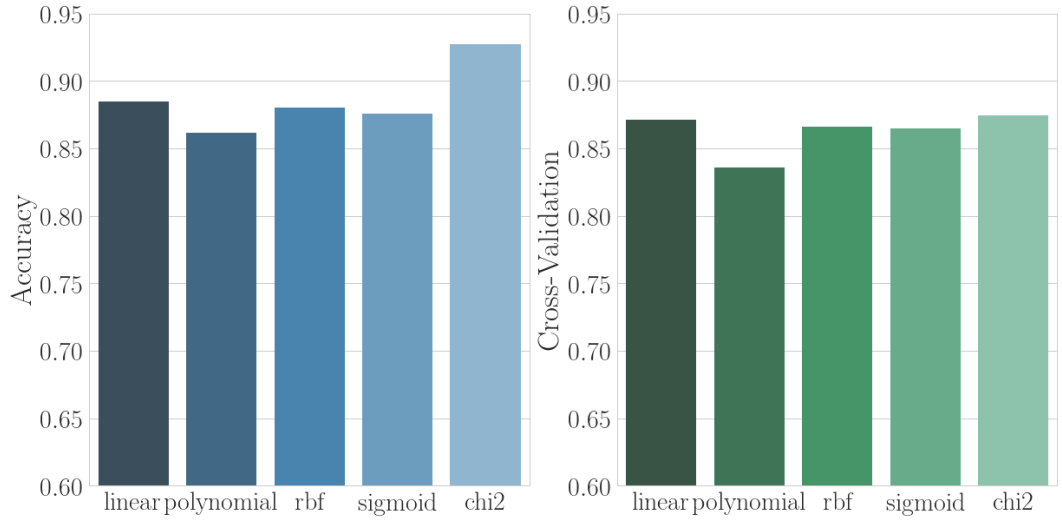

Figure 4.1: Accuracy and cross-validation scores with different SVC kernel functions. Chi-squared kernel seems to be the best choise in the context of this data.

Support Vector Classifier also requires the regularization parameter R to be specified, the default being 1.0. The accuracy scores and cross-validation scores with different parameter values obtained with chi-squared kernel can be seen from the Figure 4.2. In general, larger values of R provide better accuracy scores, but cross-validation scores remain relatively stable for values larger than 5.0. Therefore, it seems that SVC is not very sensitive to the choice of regularization parameter, as long as it is larger than 5.0. For the following results, the value of 13.0 is used for regularization parameter. Thus, the hypothesis class becomes:

$$\mathcal{H}_{\text{SVC}} = \{h \mid K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \sum_k \frac{(\mathbf{x}_{i,k} - \mathbf{x}_{j,k})^2}{\mathbf{x}_{i,k} + \mathbf{x}_{j,k}}\right), R = 13.0\}. \tag{4.5}$$

Binary classification results for the set F1, obtained with SVC, using chi-squared kernel and regularization parameter value of 13.0, are relatively similar with the ones obtained with Random Forest Classifier with the same set of features. The accuracy score of **92.738**% and cross-validation score of **87.443**% (with 5 folds) are slightly higher compared to the results obtained with Random Forest Classifier. It can be seen from the Confusion Matrix (4.6), that the model confuses 289 examples to be class 1, which actually belong to class 0, whereas the amount of false class 0 examples is now 47.

$$\mathbf{C}_{\text{SVC1}} = \begin{bmatrix} & \text{class0} & \text{class1} \\ \text{class0} & 557 & 289 \\ \text{class1} & 47 & 3734 \end{bmatrix} \tag{4.6}$$

When leaving out all examples from class 10 and repeating binary classification

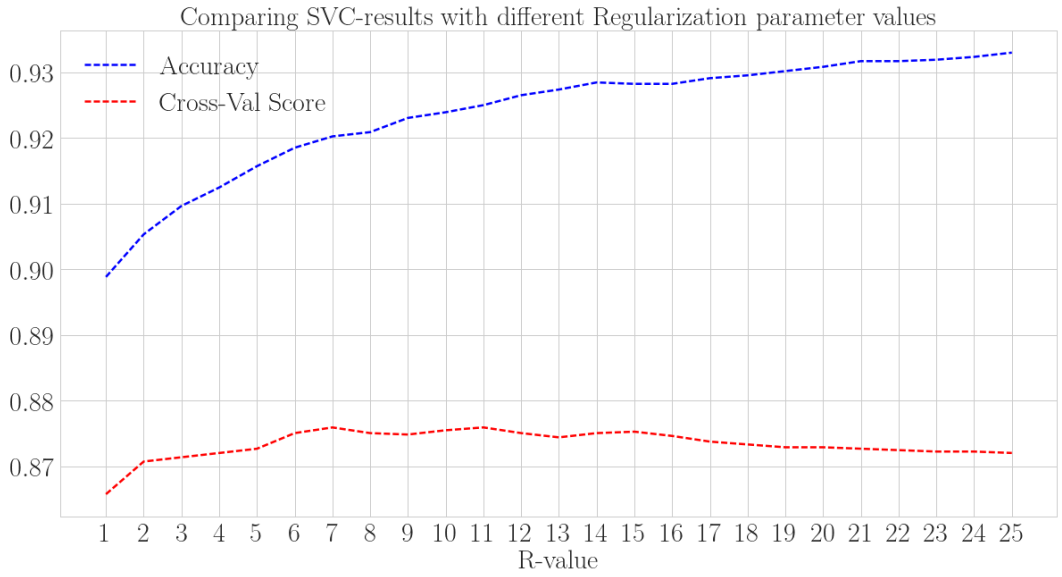

Figure 4.2: Accuracy scores and cross-validation scores with different values of regularization parameter obtained with SVC on data set F1. The algorithm is not very sensitive to the choice of regularization parameter on this data set if value larger that 5.0 is chosen.

using SVC with the parameters explained above on the set F1, again the results improve significantly, compared to the situation where class 10 examples are included. The accuracy score of **97.377**% and cross-validation score of **92.128**% are obtained.

$$\mathbf{C}_{\mathrm{SVC2}} = \begin{bmatrix} & \text{class0} & \text{class1} \\ \text{class0} & 315 & 97 \\ \text{class1} & 13 & 3768 \end{bmatrix} \tag{4.7}$$

For the multiclass case, Support Vector Classification provided accuracy of **95.421**% and cross-validation score of **91.899**% on the set F1. It can be seen from the Confusion Matrix (4.8), that SVC performs similarly to Random Forest Classifier in multiclass classification. Almost all class 1 and class 2 examples are again correctly classified, but the model confuses all other classes to be class 1. The biggest difference between Random Forest Classifier and SVC is the prediction accuracy of other classes besides 1, 2, 4 or 6, here labeled as 0. Random Forest is able to correctly classify 146 examples whereas SVC is only able to correctly classify 12 of class 0. SVC is not able to correctly classify any of the class 4 or 6 examples.

$$\mathbf{C}_{\text{SVC3}} = \begin{bmatrix} & \text{class0} & \text{class1} & \text{class2} & \text{class4} & \text{class6} \\ \text{class0} & 12 & 192 & 1 & 0 & 0 \\ \text{class1} & 4 & 3759 & 17 & 0 & 0 \\ \text{class2} & 0 & 19 & 100 & 0 & 0 \\ \text{class4} & 0 & 56 & 0 & 0 & 0 \\ \text{class6} & 0 & 32 & 1 & 0 & 0 \end{bmatrix} \tag{4.8}$$

The results obtained with SVC and chi-squared kernel on the set F1 are summarized in Table 4.2.

Table 4.2: Results obtained with SVC (chi-squared kernel) and features F1

	All classes included	Without class 10	Multiclass case
Accuracy	92.738 %	97.377 %	95.421 %
Cross-Val Score	87.443 %	92.128 %	91.899 %

4.1.3 Multiclass classification results on balanced data set

Since it is obvious from the previous classification results, that the large number of class 1 examples dominates over other classes in the classification task, it is of interest to see whether it is possible to distinguish between all the different classes using a balanced data set and the features F1. In other words, whether the data includes enough information and the correct features to actually distinguish between the classes. The challenge in this approach is that the small number of examples of some classes limits the size of the data set used for training the model. Since there are only one sample of classes 3, 5, and 8, those classes are not included. Class 10 is also not included. Hence, $Y_{\text{multi}} = \{1, 2, 4, 6, 7, 9\}$. There are 22 examples of class 7, which is the smallest limiting number when building the balanced data set and hence, 22 examples of each class are included in the training set. Thus, $n_{\text{multi}} = 22 \cdot 6 = 132$. Different from previous multiclass classification task, here all the examples are left with their original class label, and hence, there is no class for "other" examples (previously class 0). In particular, it is of interest to see whether classes 2, 4, and 6 can be distinguished from class 1. The algorithm chosen for this is Random Forest Classifier, since it seems to provide better results in the multiclass task than SVC.

$$\mathbf{C}_{\text{RF\_balanced}} = \begin{bmatrix} & \text{class1} & \text{class2} & \text{class4} & \text{class6} & \text{class7} & \text{class9} \\ \text{class1} & 20 & 0 & 0 & 1 & 1 & 0 \\ \text{class2} & 0 & 20 & 2 & 0 & 0 & 0 \\ \text{class4} & 1 & 0 & 20 & 1 & 0 & 0 \\ \text{class6} & 3 & 0 & 2 & 16 & 0 & 1 \\ \text{class7} & 1 & 0 & 1 & 0 & 20 & 0 \\ \text{class9} & 0 & 1 & 0 & 0 & 0 & 21 \end{bmatrix} \tag{4.9}$$

From the Confusion Matrix (4.9) it can be seen, that indeed, it is possible to separate the different classes from another quite accurately using balanced data set. Only two examples from class 1 are confused as class 6 or 7, and only 2 examples from class 2 are confused as class 4. Total of 16 examples of class 4 are correctly classified, while 6 examples are misclassified as class 1, class 4 or class 9. Total of 20 examples of class 7 and 21 examples of class 9 are correctly classified. However, there were only 22 examples used for each class, and hence, the generalization of this model is likely to be very bad for the whole data set. This can be seen from Table 4.3, where accuracy is relatively high, but the cross-validation score is only 51.091%. Now that only a small part of the entire data set is used to train the model, it is possible to test with entire data set to get better impression about the generalization. Indeed, testing on the entire data results in 61.408% of accuracy, which confirms the indication of cross-validation score about poor generalization. The reason for such a low percentage can be seen from Confusion Matrix (4.10). The model can actually classify all the smaller classes quite accurately, but class 1 is now confused with other classes which then results in low overall accuracy.

$$\mathbf{C}_{\text{RF\_balanced\_Test}} = \begin{bmatrix} & \text{class1} & \text{class2} & \text{class4} & \text{class6} & \text{class7} & \text{class9} \\ \text{class1} & 2248 & 54 & 451 & 282 & 644 & 101 \\ \text{class2} & 0 & 109 & 6 & 0 & 3 & 1 \\ \text{class4} & 4 & 2 & 45 & 2 & 3 & 0 \\ \text{class6} & 4 & 1 & 3 & 20 & 3 & 2 \\ \text{class7} & 1 & 0 & 1 & 0 & 20 & 0 \\ \text{class9} & 2 & 11 & 14 & 8 & 14 & 131 \end{bmatrix} \tag{4.10}$$

Table 4.3: Results of multiclass classification on balanced data set and features F1, using Random Forest Classifier.

	Balanced set	Test on entire data set
Accuracy	88.636 %	61.408 %
Cross-Val Score	51.091 %	

4.2 Clustering results

4.2.1 Determining clustering tendency

Two dimensionality reduction techniques, PCA and t-SNE, are used to evaluate clustering tendency on the set F_{og}. Transforming the data into three dimensions with PCA and into two dimensions with t-SNE makes it possible to assess the clustering tendency and internal structure in a visual manner. The visualization of the data after PCA-transformation can be seen in Figure 4.3, and the visualization after t-SNE transformation can be seen in Figure 4.4.

At least two bigger groups can be immediately detected from the Figure 4.3. In addition, there are at least two smaller but dense groups visible. Some possible outliers can be seen near to the bigger groups. In Figure 4.4, it is more difficult to see clearly separated groups of points. However, there is some internal structure visible. Based on this information, it seems that there is internal structure present in the data, and at least two but possibly more distinct groups can be detected via visual assessment.

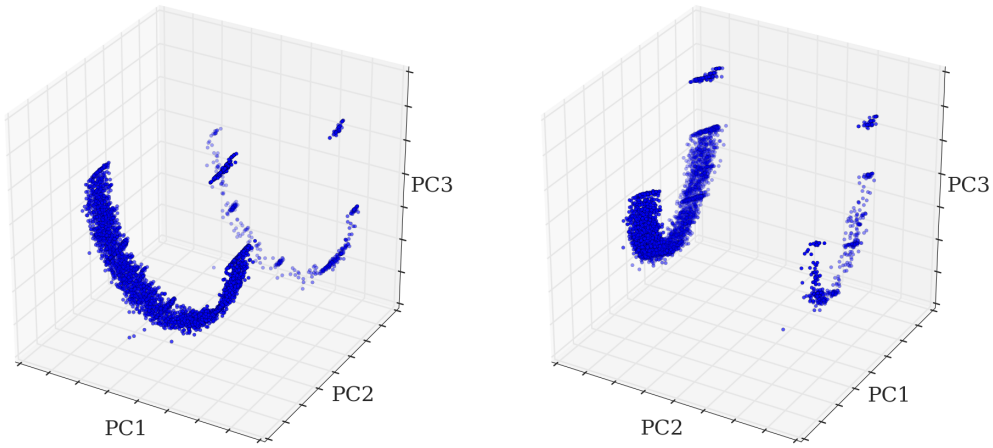

Figure 4.3: Data set F_{og} after PCA-transformation, three principal components remaining. Two bigger groups and at least two smaller groups are clearly visible.

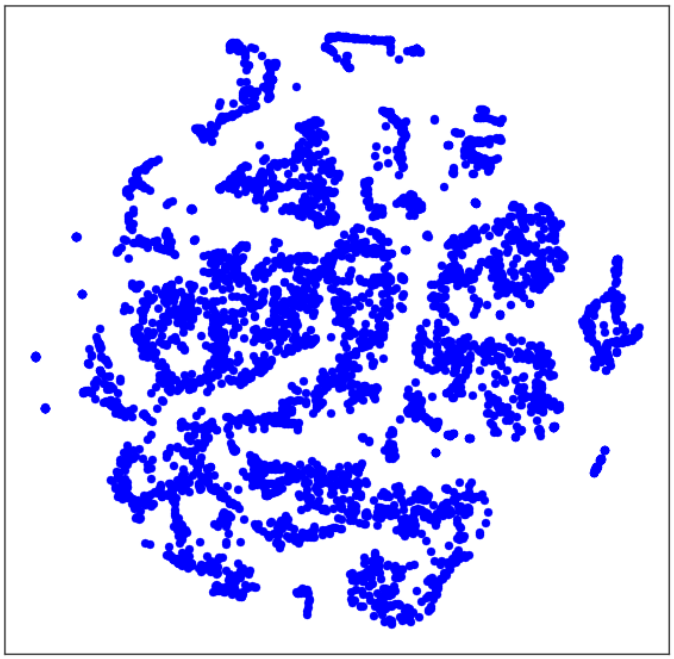

Figure 4.4: Data set F_{og} after t-SNE transformation into two dimensions. Here, it is more difficult to see clearly separated groups of points.

4.2.2 Internal evaluation

Silhouette scores are computed using three different algorithms, changing parameters of the algorithm such that the obtained number of clusters changes as well. For k-means, the number of clusters must be defined as a parameter of the algorithm, which makes it convenient for computing the silhouette scores with many different numbers of clusters. The results can be seen from Figure 4.7. With k-means, the best silhouette score is obtained when $k = 3$. Another peak in the score can be seen at $k = 6$. After k-values larger than 7, silhouette scores remain relatively stable.

The process is then repeated with Affinity Propagation and HDBSCAN. For Affinity Propagation, the number of clusters can not be specified, but the parameter *preference* can be varied in order to achieve different groupings for comparison. The results can be seen from Figure 4.7. Not a lot of variation can be seen in the silhouette scores obtained with Affinity Propagation, and the highest score is obtained at $k = 6$. However, since the number of clusters can not be specified directly, no clustering with less than 6 clusters was achieved. This makes it difficult to compare the results with the other two algorithms.

For HDBSCAN, the algorithm determines the number of clusters automatically similar to Affinity Propagation. However, the parameter *min_cluster_size* can be varied in order to obtain different groupings with different k-values. The results can be seen form Figure 4.7. In contrast to the clustering obtained with k-means, with

HDBSCAN the best silhouette score is reached when $k = 2$, which is the minimum valid value. Note that the silhouette score is lowest with $k = 6$, whereas using k-means, $k = 6$ provides higher silhouette score than $k = 5$ or $k = 7$. For all of these three algorithms, silhouette scores remain relatively stable for cluster numbers larger than 7.

Observations about how well the clustering results fit the data can be made based on visual comparison of distinct clustering results on PCA-transformed data. In Figure 4.5 two distinct clustering results obtained with k-means and HDBSCAN are compared. It can be seen, that HDBSCAN provides a clustering that captures the internal structure better than the one obtained with k-means when $k = 3$. As mentioned above, no clustering with less than 6 clusters was achieved using Affinity Propagation. Hence, it is not possible have a comparison result using AP where $k = 3$. In Figure 4.6, distinct clustering results for $k = 6$ using all the three algorithms are compared. It can be seen that the results of k-means and Affinity Propagation are relatively similar. HDBSCAN provides again such result that manages to capture the internal structure well compared to the other two algorithms.

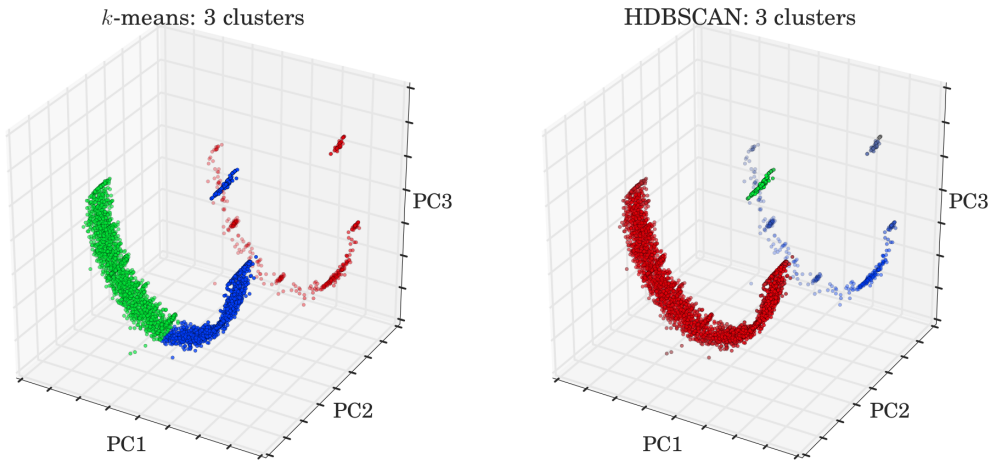

Figure 4.5: Comparison between two clustering results where number of clusters is 3. Distinct clusters are marked with different colors. First labeling is obtained with *k-means* and second with *HDBSCAN* on F_{og}.

4.2.3 External evaluation

Next it is of interest to compare the obtained cluster labels with correct class labels in order to explore whether there is something in common with the internal structure of the data and human-assigned class labels. The data set used here is F_{og}. External evaluation metrics, such as homogeneity, completeness, ARI, and AMI, are applied.

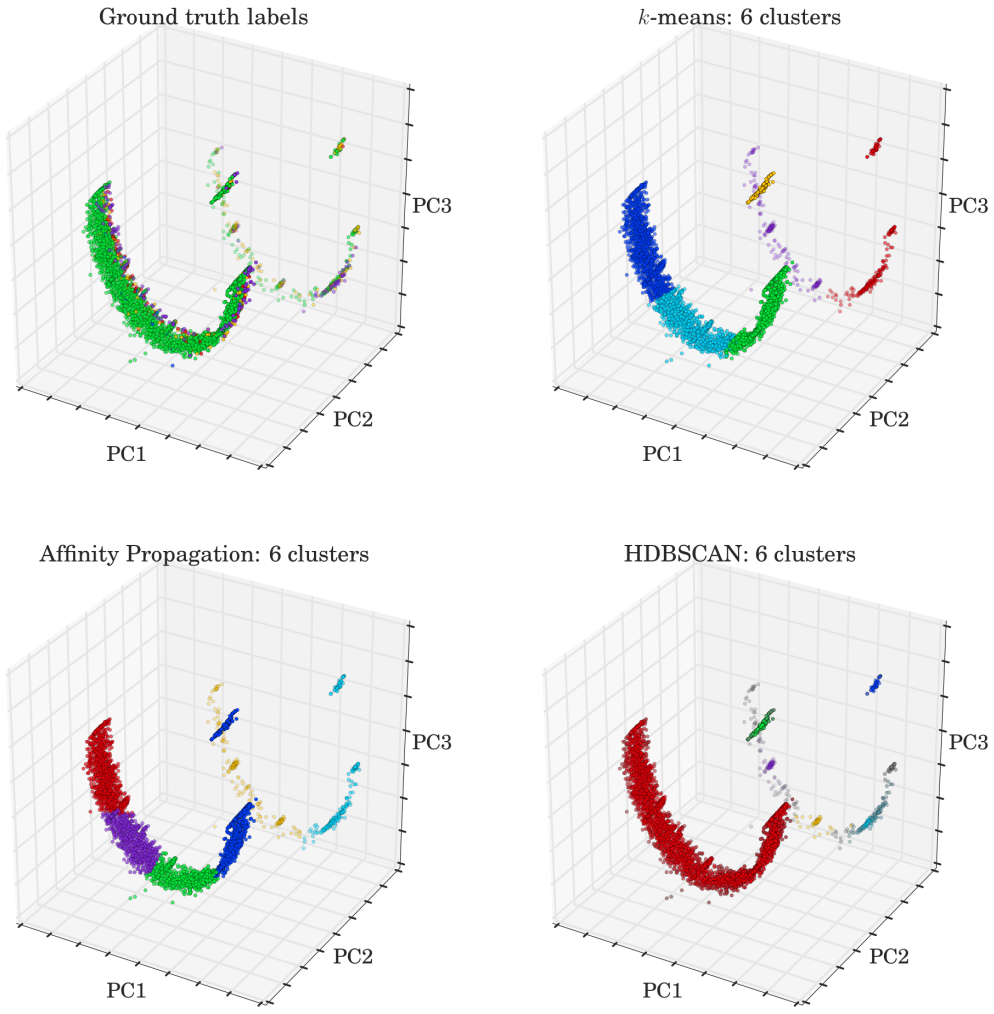

Figure 4.6: Comparison between ground truth labels and clustering results where $k = 6$ and data set F_{og}. Distinct clusters are marked with different colors.

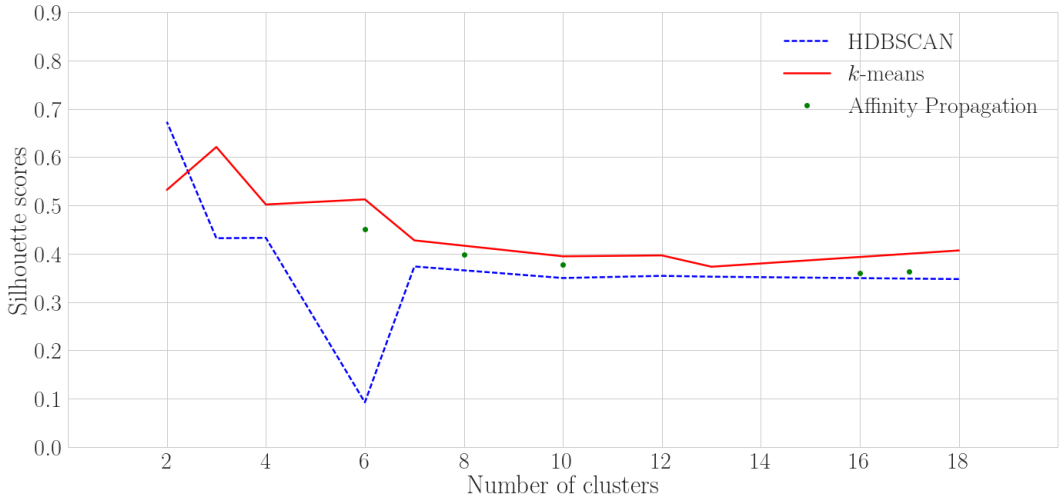

Figure 4.7: Silhouette analysis using *k-means*, *Affinity Propagation*, and *HDBSCAN* clustering on F_{og}. Best silhouette scores are obtained with $k < 7$ with k-means and Affinity Propagation, and $k < 4$ with HDBSCAN. After $k > 7$, silhouette score remain relatively stable for all three algorithms.

In addition, visual assessment is used to evaluate goodness and fit of the obtained clustering. The results obtained with these metrics can be seen in Figure 4.9, Figure 4.10, and Figure 4.11, using different clustering algorithms.

For k-means, homogeneity is the only score reaching larger values than 0.1 with any number of clusters. The overall trend is that homogeneity is increasing as the number of clusters increases, which is reasonable, since a cluster satisfies homogeneity if all of its points are members of a single class. All other external metrics remain close to zero, which indicates almost random clusters with reference to the real labels. For Affinity Propagation, situation is very similar, homogeneity being the only metric reaching larger values than 0.1. Based on this information, it is clear that neither k-means nor Affinity Propagation are able to produce such clustering that has something in common with the human-assigned class labels.

From Figures 4.9, 4.10, and 4.11 it can be seen, that HDBSCAN seems to provide such clustering that best fits the correct class labels. The best external evaluation results are obtained with $k = 6$, in contrast to the results obtained with silhouette analysis, where 6 seemed to be the worst choice when determining the fitting number of clusters. At 6 clusters, ARI is approximately 0.33 and AMI 0.15. These are still relatively low scores, taking into account that the maximum score that implies similar labelings is 1.0.

In Figure 4.8, different cluster labelings are compared with the ground truth labeling via t-SNE visualization. Similar comparison can also be seen in Figure 4.6 using PCA-visualization. Since the internal structure is not corresponding with the ground truth labels, it is reasonable, that the different clustering algorithms

provide such results that are not able to separate the different classes into distinct clusters. This difference between internal structure and external labels, which was indicated by the external metrics, can easily be detected from Figures 4.8 and 4.6. One interesting detail in the Figure 4.8 is that class 2 examples correspond to the red points in the left upper corner plot, which are clearly separated as their own group. It is notable, that none of these algorithms are able to separate this clearly separated group of points from other groups.

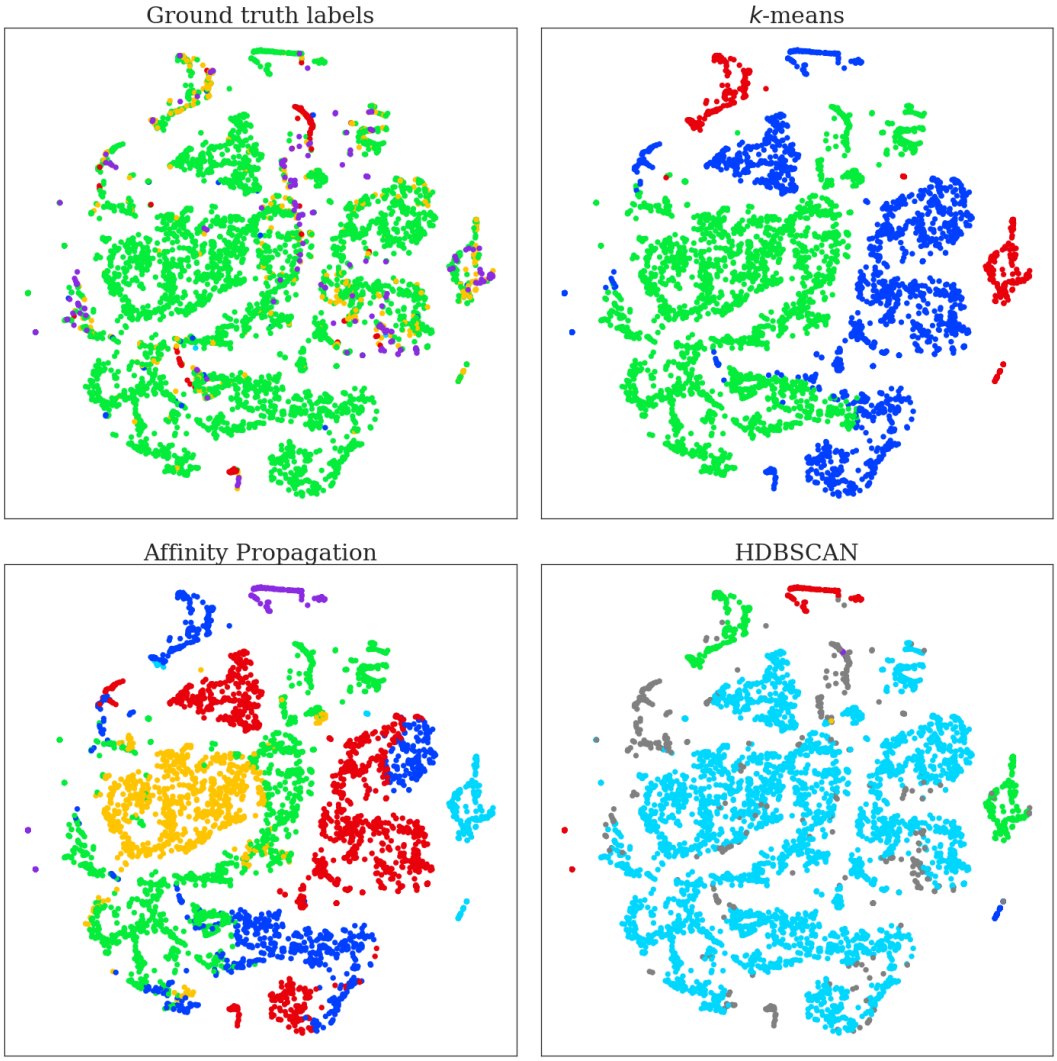

Figure 4.8: Visualizations of different cluster labelings on F_{og}. Left upper corner; ground truth labeling, right upper corner; k-means with 3 clusters, left lower corner; Affinity Propagation with 8 clusters, right lower corner; HDBSCAN with 5 clusters.

4.3 Classification results with PFH-features

Using data set combined from all the features including PFH-features and choosing the best features with feature importance, classification task is repeated in order

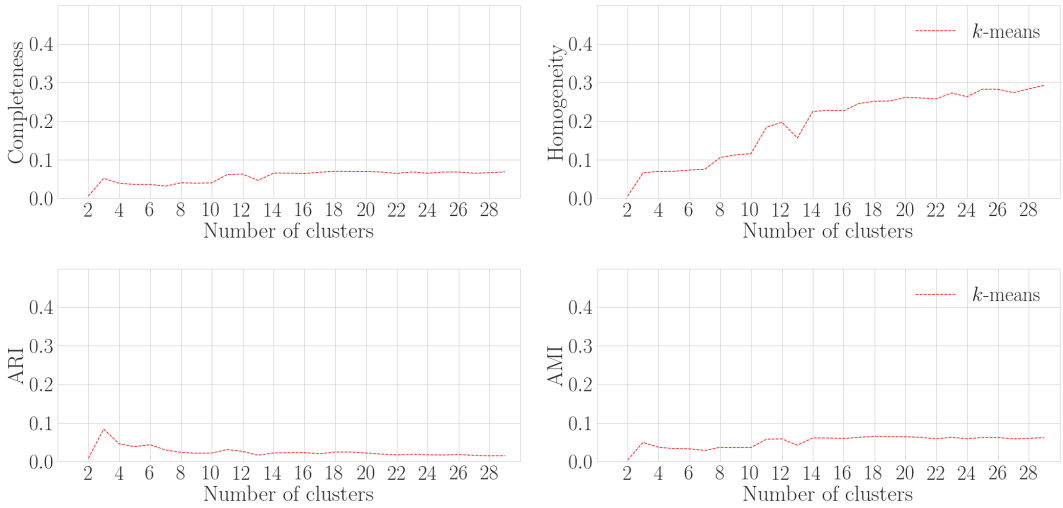

Figure 4.9: External evaluation metrics using *k-means* on F_{og}. Left upper corner; completeness score, right upper corner; homogeneity score, left lower corner; Adjusted Rand Index, right lower corner; Adjusted Mutual Information.

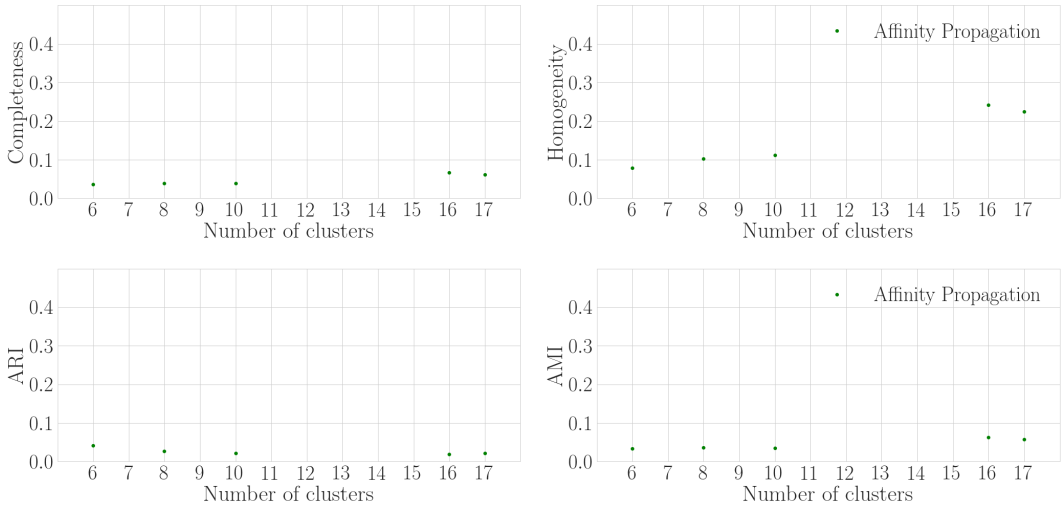

Figure 4.10: External evaluation metrics of *Affinity Propagation* on F_{og}. Left upper corner; completeness score, right upper corner; homogeneity score, left lower corner; Adjusted Rand Index, right lower corner; Adjusted Mutual Information.

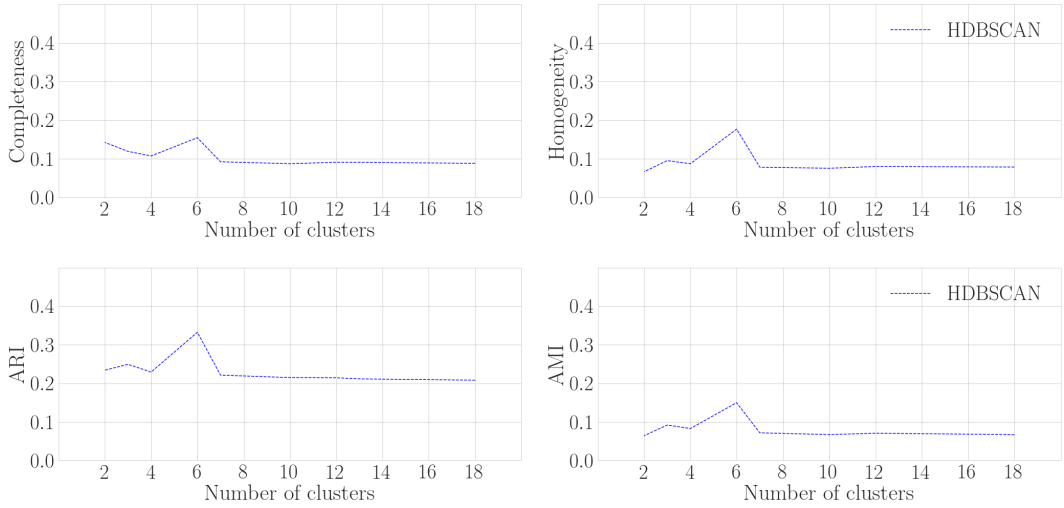

Figure 4.11: External evaluation metrics of *HDBSCAN* on F_{og}. Left upper corner; completeness score, right upper corner; homogeneity score, left lower corner; Adjusted Rand Index, right lower corner; Adjusted Mutual Information.

to be able to compare the results with original ones. The results are summarized in the following sections, divided again into results obtained with Random Forest Classifier and SVC. The last section is again concerning the multiclass classification on balanced data set with new features in order to compare these results as well.

4.3.1 Selecting the best features

After computing the PFH-describtors and combining the 125 features with the original and gaussian features (F1), resulting number of features is $d = 159$. It is not necessary nor beneficial to use all these features since high dimensionality slows down the algorithms but does not usually improve the results significantly. For example, t-SNE slows down very quickly when the dimensionality is increasing. Hence, to get better performance and reasonable running times, only the best and most important features in the context of classification task are chosen using Feature Importance of Random Forest Classifier. In Scikit-learn library's feature selection, there is a class called SelectFromModel, which is a meta-transformer for selecting features based on importance weights [24]. The technique of ranking the variables by their importance in a natural way was originally described in [1]. It takes as parameters the chosen estimator (here, Random Forest Classifier) and the *threshold*, which refers to the threshold value of feature importance. Those features whose importance is greater or equal to the threshold are kept while the others are dropped [24]. More detailed description of Feature Importance and SelectFromModel can be found in [24]. The threshold chosen is 0.009, and the resulting set of features can be seen in Table 4.4. This set of features, referred to as F2, is used to obtain the following results. Note

that for 6 examples in the data set, it was not possible to compute point normals because there were too few points in the point cloud data. Hence, it was also impossible to compute the PFH descriptors for these examples. Therefore, these examples were dropped from the combination data set F2 and the total number of examples for F2 is $n_{\mathrm{F2}} = 4621$.

Table 4.4: Best features based on Feature Importance of Random Forest, using threshold of 0.009, later referred to as F2.

1. PFH: '5'
2. PFH: '8'
3. PFH: '9'
4. PFH: '15'
5. PFH: '19'
6. PFH: '27'
7. PFH: '28'
8. PFH: '45'
9. PFH: '48'
10. PFH: '52'
11. PFH: '121'
12. kernel-channel
13. blob-perimeter
14. blob-rectangularity
15. reconstruction-zmin
16. reconstruction-zmax
17. lw-ratio
18. hd-ratio
19. z_min / z_rms
20. z_max / z_rms

4.3.2 Random Forest Classifier

Applying Random Forest Classifier to the set F2 and including the examples from all classes, the accuracy of **92.599**% and cross-validation score of **87.232**% are obtained for binary classification, within $\mathcal{H}_{\mathrm{RF}}$.

$$\mathbf{C}_{\mathrm{RF\_New1}} = \begin{bmatrix} & \text{class0} & \text{class1} \\ \text{class0} & 577 & 263 \\ \text{class1} & 79 & 3702 \end{bmatrix} \tag{4.11}$$

When dropping all class 10 examples and repeating classification on the set F2, accuracy of **96.991**% and cross-validation score of **92.117**% are obtained for binary classification.

$$\mathbf{C}_{\text{RF\_New2}} = \begin{bmatrix} & \text{class0} & \text{class1} \\ \text{class0} & 291 & 118 \\ \text{class1} & 8 & 3770 \end{bmatrix} \tag{4.12}$$

When changing the labeling again into multiclass labeling as explained above, the accuracy obtained wit Random Forest Classifier is **96.011**% and cross-validation score is **93.958**%. However, from the Confusion Matrix (4.13) it can be seen that, similar to the original results, classes 0, 1 and 2 are mostly correctly classified, whereas classes 4 and 6 are again confused as class 1 examples. There are, however, 3 more examples correctly classified as class 4 examples compared to the original situation.

$$\mathbf{C}_{\text{RF\_New3}} = \begin{bmatrix} & \text{class0} & \text{class1} & \text{class2} & \text{class4} & \text{class6} \\ \text{class0} & 138 & 64 & 0 & 0 & 0 \\ \text{class1} & 3 & 3772 & 3 & 0 & 0 \\ \text{class2} & 1 & 14 & 104 & 0 & 0 \\ \text{class4} & 0 & 49 & 0 & 6 & 0 \\ \text{class6} & 1 & 32 & 0 & 0 & 0 \end{bmatrix} \tag{4.13}$$

The results using Random Forest Classifier on the set F2 are summarized in the Table 4.5.

Table 4.5: Results obtained with Random Forest Classifier and the data set F2.

	All classes included	Without class 10	Multiclass case
Accuracy	92.599 %	96.991 %	96.011 %
Cross-Val Score	87.232 %	92.117 %	93.958 %

4.3.3 Support Vector Classification

The accuracy scores and cross-validation scores with different regularization parameter values obtained with chi-squared kernel are again evaluated in order to see how the results are changing. The results can be seen from Figure 4.12. Accuracy scores are again increasing with larger R-values, but the cross-validation scores remain relatively stable for R-values larger than 4.0 and smaller than 16.0. The value chosen for comparison is again $R = 13$.

The accuracy obtained while including all the classes and applying SVC on the set F2 is **90.348**% and the cross-validation score is **87.687**%.

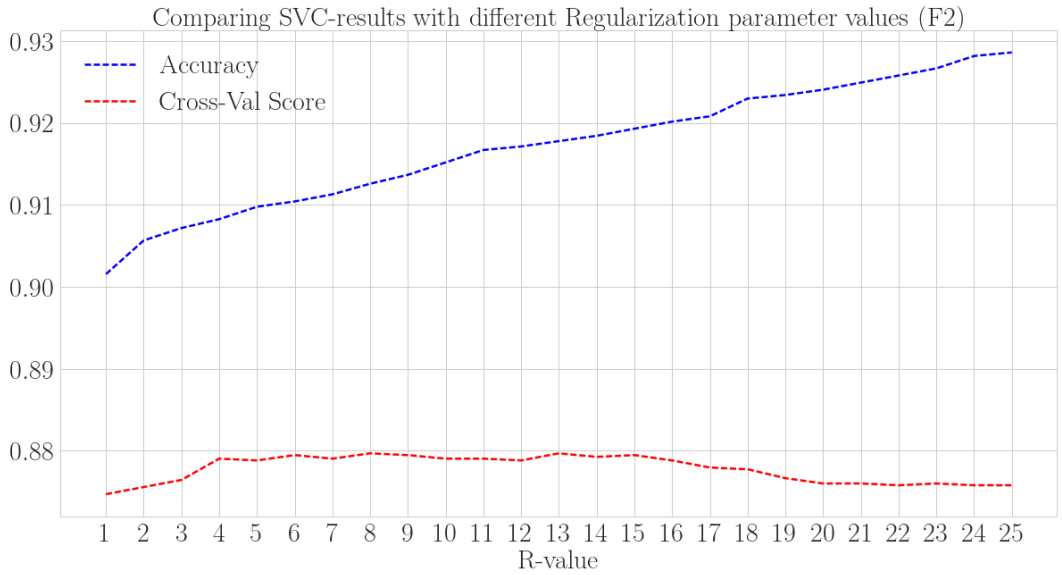

Figure 4.12: Accuracy and cross-validation scores with different values of regularization parameter obtained with SVC on F2. Accuracy scores are increasing when R is increasing, but cross-validation scores remain relatively stable when $4 < R < 16$.

$$\mathbf{C}_{\text{SVC\_New1}} = \begin{bmatrix} & \text{class0} & \text{class1} \\ \text{class0} & 596 & 244 \\ \text{class1} & 202 & 3579 \end{bmatrix} \tag{4.14}$$

When dropping all class 10 examples, the accuracy obtained with SVC is **95.988**% and cross-validation score is **91.735**% on the set F2.

$$\mathbf{C}_{\text{SVC\_New2}} = \begin{bmatrix} & \text{class0} & \text{class1} \\ \text{class0} & 261 & 148 \\ \text{class1} & 20 & 3758 \end{bmatrix} \tag{4.15}$$

Finally, for the multiclass case, the accuracy obtained with SVC and the set F2 is **94.793**% and the cross-validation score **91.902**%. However, from the Confusion Matrix (4.16) it is again clear, that the high accuracy is due to high number of correctly predicted class 1 examples, whereas examples from classes 4 and 6 are incorrectly classified as class 1.

$$\mathbf{C}_{\text{SVC\_New3}} = \begin{bmatrix} & \text{class0} & \text{class1} & \text{class2} & \text{class4} & \text{class6} \\ \text{class0} & 107 & 95 & 0 & 0 & 0 \\ \text{class1} & 9 & 3762 & 7 & 0 & 0 \\ \text{class2} & 1 & 18 & 100 & 0 & 0 \\ \text{class4} & 0 & 55 & 0 & 0 & 0 \\ \text{class6} & 1 & 31 & 1 & 0 & 0 \end{bmatrix} \tag{4.16}$$

Table 4.6: Results obtained with SVC and the feature set F2.

	All classes included	Without class 10	Multiclass case
Accuracy	90.348 %	95.988 %	94.793 %
Cross-Val Score	87.687 %	91.735 %	91.902 %

Table 4.7: Summary of all classification results with both data sets F1 and F2.

	All classes	**Without class 10**	**Multiclass**
Data set F1			
Random Forest Classifier			
Accuracy	92.695 %	96.780 %	95.969 %
Cross-Val Score	86.882 %	91.770 %	94.086 %
SVC			
Accuracy	92.738 %	97.377 %	95.421 %
Cross-Val Score	87.443 %	92.128 %	91.899 %
Data set F2			
Random Forest Classifier			
Accuracy	92.599 %	96.991 %	96.011 %
Cross-Val Score	87.232 %	92.117 %	93.958 %
SVC			
Accuracy	90.348 %	95.988 %	94.793 %
Cross-Val Score	87.687 %	91.735 %	91.902 %

4.3.4 Multiclass classification on balanced data set with new features

The generation of balanced data set is repeated with the data set F2 in order to compare the results. It is of interest to see whether the combination data set helps to separate between the classes more accurately than the original one. The algorithm used is again Random Forest Classifier and the number of examples from each class remains the same. From the Confusion Matrix (4.17) it can be seen, that the results are very similar to the original situation. The algorithm is able to distinguish

between all the classes quite accurately. Only minor variations in the confused examples can be seen in the confusion matrix. From Table 4.8 it can be seen, that the accuracy is slightly higher than with the original data set. However, cross-validation score is even lower than before, and testing with the entire data set confirms this observation of poor generalization.

$$\mathbf{C}_{\text{RF\_balanced\_New}} = \begin{bmatrix} & \text{class1} & \text{class2} & \text{class4} & \text{class6} & \text{class7} & \text{class9} \\ \text{class1} & 20 & 0 & 1 & 1 & 0 & 0 \\ \text{class2} & 0 & 20 & 2 & 0 & 0 & 0 \\ \text{class4} & 0 & 0 & 21 & 1 & 0 & 0 \\ \text{class6} & 2 & 0 & 1 & 19 & 0 & 0 \\ \text{class7} & 1 & 0 & 3 & 0 & 18 & 0 \\ \text{class9} & 1 & 0 & 0 & 0 & 0 & 21 \end{bmatrix} \tag{4.17}$$

Table 4.8: Results of multiclass classification on balanced data set using Random Forest Classifier and feature set F2.

	Balanced set	Test on entire data set
Accuracy	90.152 %	57.839 %
Cross-Val Score	44.147 %	

4.4 Clustering with PFH-features

4.4.1 Clustering tendency

After including new features and choosing the best ones using feature importance, PCA-transformation into three dimensions and t-SNE transformation into two dimensions are repeated in order to evaluate clustering tendency of the data set F2. From Figure 4.13, it can be seen that the plot is now different compared to the original one. The points are located into more dense groups, and only two distinct groups can clearly be seen in the visualization. From Figure 4.14, it is again more difficult to see clear separation of points into distinct groups. However, at least one bigger group of points is visible. In addition, some smaller and less dense groups can be seen in the plot. Based on these visualizations, it seems that there is less internal structure present in the new data set compared to the original one, without any reference to the ground truth labeling.

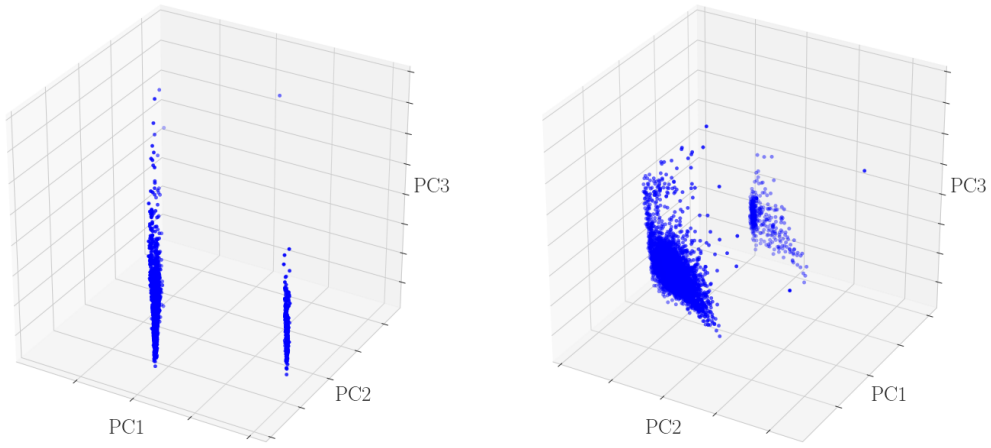

Figure 4.13: Visualization of PCA-transformed data with new features. Only two dense groups of points are visible.

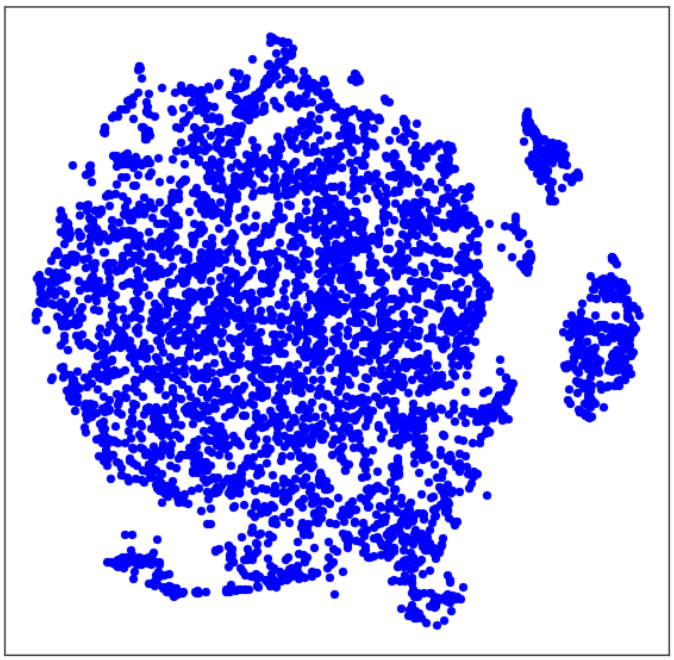

Figure 4.14: Visualization of t-SNE transformed data with new features. Compared to original t-SNE transformation, less internal structure is visible.

4.4.2 Internal evaluation

Since PCA-visualization (Figure 4.13) on the set F2 is very dense, it is difficult to see the differences between cluster labelings. Therefore, PCA is not used in this section, but only silhouette analysis and t-SNE plots are used for internal evaluation of new data set.

In Figure 4.15, silhouette scores obtained with k-means, Affinity Propagation, and HDBSCAN are presented. HDBSCAN provided only two clusters with every run, and hence, there is only one silhouette score to compare with the other two algorithms. This result agrees with the indications of PCA-transform providing only two dense groups. For k-means and Affinity Propagation, lower number of clusters provide better silhouette scores, which seems reasonable for the same reason. For more than 2 clusters, highest silhouette scores for k-means are obtained with 3, 7, and 9 clusters, whereas for AP, with 6 and 11 clusters.

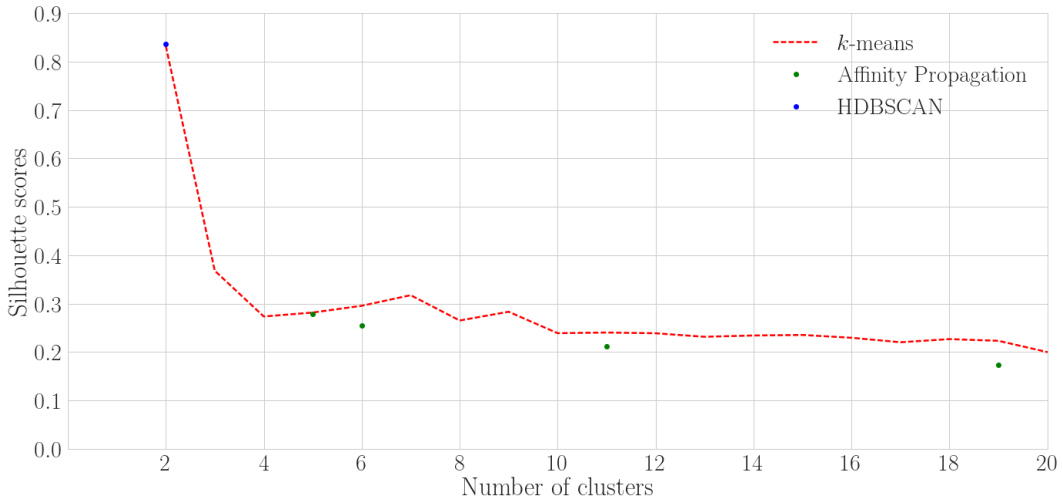

Figure 4.15: Silhouette analysis on k-means, Affinity Propagation, and HDBSCAN on the data set F2.

4.4.3 External evaluation

When applying external evaluation metrics on the new data set with k-means and Affinity Propagation, the external evaluation metrics provide very similar scores for both algorithms. HDBSCAN provided only one result with two clusters, and hence, it is not comparable with the two other algorithms. Homogeneity scores are again increasing when the number of clusters is increasing with both two algorithms. Interesting is, that for new data set F2, completeness, AMI, and ARI produce slightly better scores with both k-means and AP. However, all these three scores are still relatively low, indicating that there is not much similarity between cluster labels

and the ground truth labels.

In Figure 4.18, different cluster labelings are compared to the ground truth labeling. Interesting is, that even though there seems to be less internal structure and less clear groups visible in t-SNE visualization, now the different classes are more concentrated on distinct areas. In particular, there is a large area of class 1 examples (green) and the areas where most of other classes are located can be separated from this bigger area. Therefore it is possible to divide the data set into clusters that better correspond to the ground truth labels. Note, that both AP and k-means provide very similar cluster labelings on the new data set. Both are able to distinguish between most examples belonging to class 2 (red) and other classes, and both are able to distinguish between areas that are mostly class 1 (green) and areas where most of other classes are located. Note also, that the larger area of class 1 examples is divided into 3 bigger areas in both labelings obtained with AP and k-means.

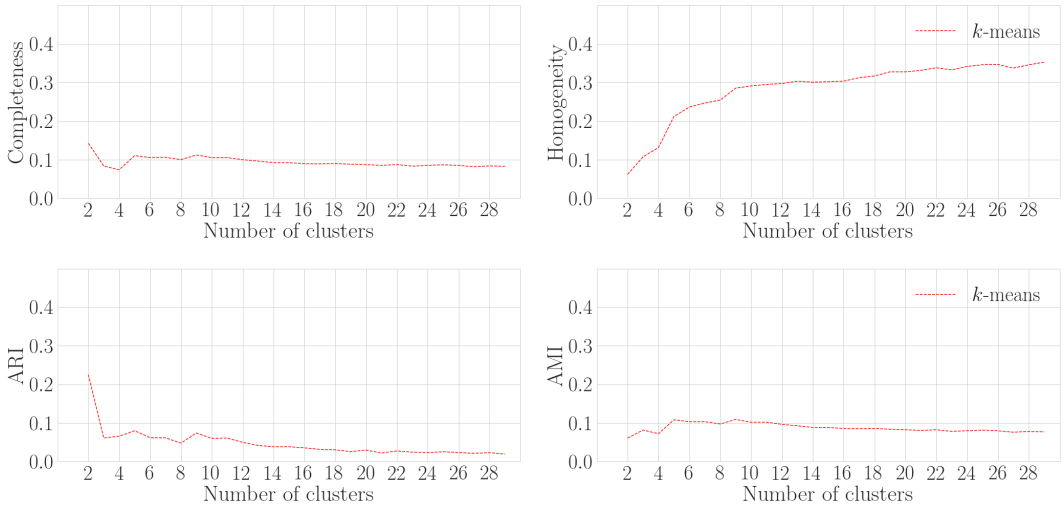

Figure 4.16: External evaluation on k-means clustering and new data set.

4.5 Discussion

Clustering

With the data set F1, there seems to be internal structure present, and separate groups of points are clearly visible. It is possible to distinguish well separated clusters without reference to ground truth labels. However, with reference to the ground truth labeling, it becomes obvious that the internal structure does not correspond to the externally known classes. Clustering algorithms usually generate clusters of approximately equal size, which makes the correspondence between real class labels and resulting cluster labeling poor in the context of highly unbalanced data.

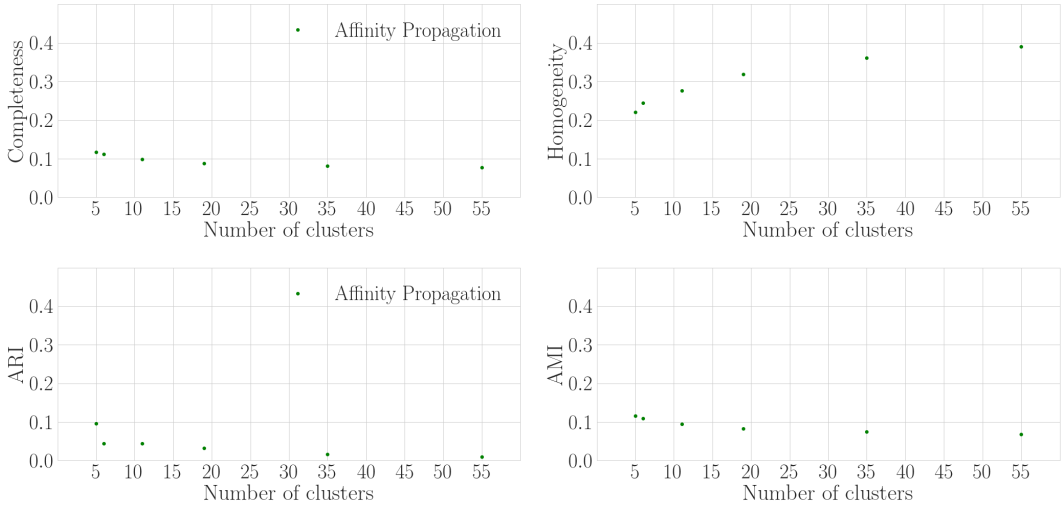

Figure 4.17: External evaluation on Affinity Propagation clustering and new data set.

After generating the new data set F2 based on feature importance, there seems to be less internal structure present compared to the original situation, since not so many distinct sets of points are visible. However, with reference to the ground truth labeling, the clustering results seem to be slightly better, because most of the class 1 samples are located in the same area, and most of class 2 samples are also located in a separate area. It is also possible to distinct the area, where most of the other classes besides 1 and 2 are located. External evaluation metrics also confirm this finding, because the scores of all 4 evaluation metrics are just slightly higher than for the original data set. With reference to the ground truth labeling, it is actually reasonable to have only one bigger group of points and a couple of smaller ones, since class 1 covers over 81 % of the data set.

It is obvious that high external evaluation scores are hard to reach with this particular data, as there are total of 10 classes, and some of these classes only contain one or two examples. Such clustering is not likely or even possible to achieve with any kind of algorithm. In addition, there seems to be more than just one subgroup in class 1 based on clustering results, which is an interesting finding, that needs further research in order to be confirmed. Furthermore, with this data, it is only possible to characterize the shape of the surface of the defect. Hence, it is not always clear, which characteristics define which classes. This is better explained through visualization in Appedix A: Figure A.3. Each of these defects are partly going below the surface and partly above the surface, but not all of them belong to the same class. This can happen due to an error of the human inspector, but also due to some characteristics not visible in the surface plot. This is a big challenge for both clustering and classification algorithms, since very similar instances based on surface shape can actually belong to separate classes.

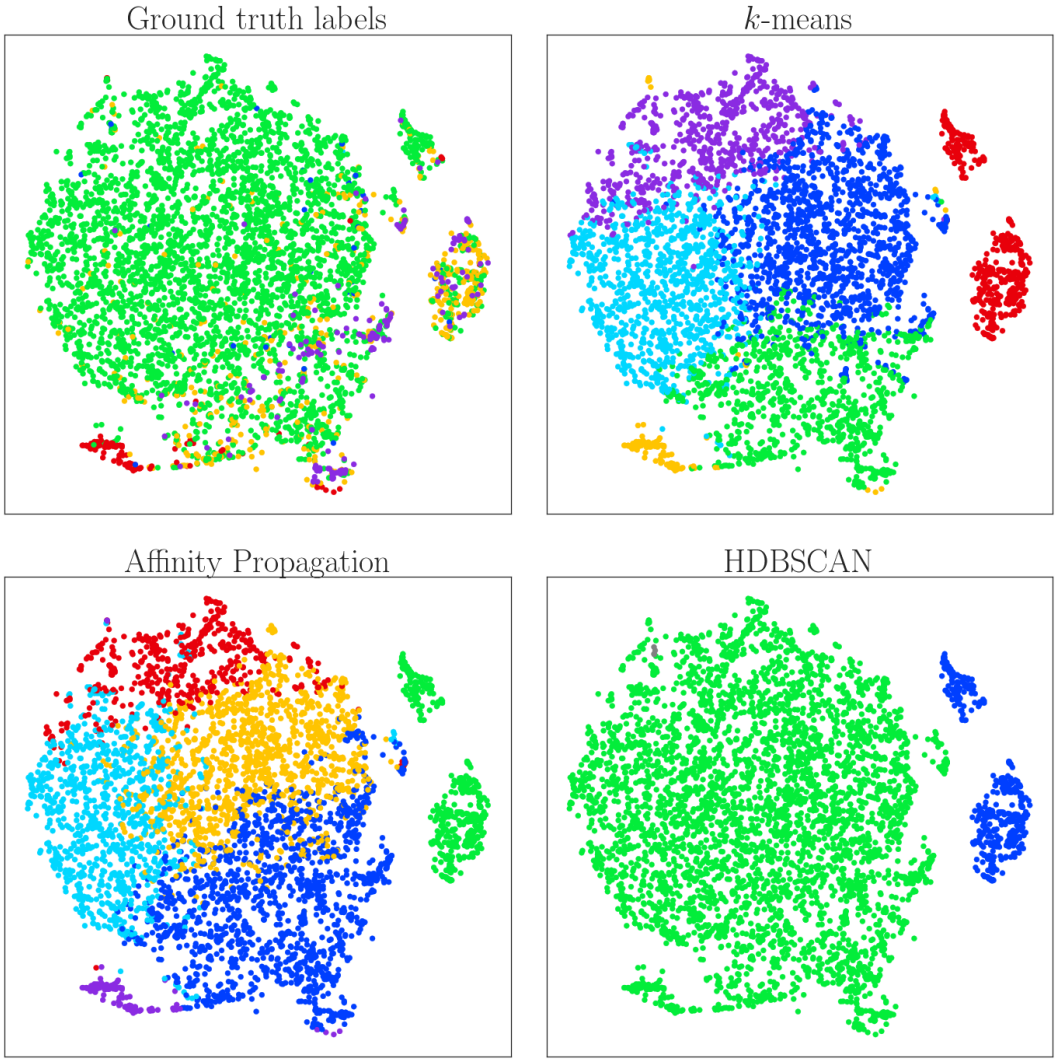

Figure 4.18: Visualizations of different cluster labelings on the new data set F2 using t-SNE. Left upper corner; ground truth labeling, right upper corner; k-means with 6 clusters, left lower corner; Affinity Propagation with 6 clusters, right lower corner; HDBSCAN with 2 clusters.

Classification

Classification results are relatively accurate with both algorithms, since the data set is unbalanced and class 1 dominates the classification task. Class 10 plays a significant role in classification accuracy, since without class 10 samples the results are remarkably better. This seems reasonable, since class 10 consists of unlabeled examples and there is no actual information available about the class, which then confuses the model. For multiclass case, accuracy remains relatively high but this is also due to the large amount of class 1 examples being correctly classified. Classes 4 and 6 are totally confused as class 1 examples, but they are so few that it does not affect the accuracy significantly. In real-life applications, however, it would be important to be able to distinguish between the smaller and more rare cases as well. Random Forest Classifier seems to perform better than SVC in multiclass case, but otherwise the two algorithms perform very similarly.

The classification results obtained with new data set F2 remain very similar to the original situation. Even though the PFH-features seem to be useful in the clustering task, the classification results do not improve when the new features are included. When taking a closer look to the results provided by Random Forest Classifier in the multiclass situation and comparing class 2 examples which are incorrectly classified as class 1 or class 0 to the examples that are correctly classified, the results are interesting. It seems that those defects, which are not only going below the surface but have also one or more "bumps" above the surface, are likely to be classified as class 1 or class 0 rather than class 2. The examples can be seen in Appendix A: Figure A.1 and Figure A.2. This again demonstrates the challenge related to reflection data, but also builds trust on the model, since the misclassifications are reasonable in some sense.

The biggest issue in the multiclass situation for both data sets is that classes 4 and 6 are totally confused as class 1 samples. This also seems quite reasonable after browsing through samples of these classes. In most cases, there is something above the surface similar to class 1, but the number and shape of "bumps" is varying inside the classes as well as between the classes. The algorithm is then classifying all the examples of class 4 and 6 as class 1, since there are so many class 1 examples, that it is dominating over other classes, even though there actually are some differences in the surface shapes. In order to be able to correctly classify classes 4 and 6 (and possibly 3, 7, and 8), more examples from these classes are needed for training and testing.

It is also possible to generate balanced data sets (equal number of samples of each class) in order to avoid one class dominating over others. However, in this particular case it turned out to be not very beneficial to use this classifier to perform classification on the entire data for two reasons. First, as discussed already, there

are still only 22 examples from class 7, 33 examples of class 6 and 56 of class 4. This limits the number of samples for training quite dramatically, which yields poor generalization. Second, the data set is unbalanced for a reason: In real life, class 1 is also the most common defect class. If the model is trained on balanced data set, the results for a real situation are not very good, since class 1 samples are then confused often as some other class. This can be seen in Confusion Matrix (4.10). Hence, in this case, it is actually beneficial to prefer class 1 over others, in order to reach higher accuracy in real life situations.

Now that it is confirmed, that the model trained on the balanced set can actually differentiate between all the classes quite well, one possible approach to achieve better accuracy in the real situation would be to use two different models sequentially: First use binary classification to train a model which can differentiate between class 1 and others, similar to the classifiers introduced first in this thesis. This step would help getting more balanced set for further multiclass classification. Next, take only those samples, which were not classified as class 1 in the previous step and use the classifier trained on balanced data set to distinguish between the remaining classes. This would solve the problem of class 1 dominating over other classes in the multiclass case and enable accurate classification for both class 1 and the smaller classes. This approach still needs some further research in order to have more information about the performance.

5. CONCLUSIONS

In this chapter, the conclusions based on findings and results explained in previous chapter are presented in order to answer the research questions in Chapter 3. In addition, some extensions for future work are suggested.

Research question 1: *Does internal structure exist in the data, and if so, does it have something in common with known class labeling?*

Based on silhouette analysis, it can be determined that internal structure does exist in the data set. For the original features F_{og}, silhouette scores slightly higher than 0.6 are obtained with k-means and HDBSCAN, and with Affinity Propagation, silhouette scores of 0.45 at highest. It seems, that with the original features, 2-4 clusters provide the highest silhouette scores with k-means and HDBSCAN. With Affinity Propagation, the lowest possible number of clusters obtained by changing the preference was 6, which also provides the highest silhouette score for Affinity Propagation. To conclude, the original data set has 2-4 relatively well separated clusters. When the number of clusters is increased, the resulting clusters are not as well separated, but not completely overlapping either.

Using the combination data set F2 with the new features, silhouette scores higher that 0.8 are obtained for two clusters. Furthermore, HDBSCAN only provided results having two clusters. It is a density-based clustering method and hence, this result implies that there are only two dense groups of points, which are then surrounded by a region of low density. With this data set, silhouette scores are decreasing faster as the number of clusters increases for AP and k-means. To conclude, the new data set has 2 relatively well separated clusters, but if the number of clusters is increased, the resulting clusters are not well separated at all. However, silhouette score only evaluates the distances within clusters versus the distances between separate clusters. For this reason, visual assessment was also used to confirm these findings.

Based on visual assessment via PCA and t-SNE dimensionality reduction on F_{og}, it can be confirmed, that some internal structure does indeed exist. PCA-visualization shows 2-4 dense groups of points. With t-SNE it is more difficult to define the number of groups present in the data, but some structure is clearly visible. Using the combination data set F2 with new features, PCA-visualization becomes more concentrated, showing only two separate, very dense groups of points.

Furthermore, t-SNE plot shows only one larger group and possibly one or more smaller ones. These results agree with the observations based on silhouette analysis: There are less internal structure present when using the combination data set, but with the minimum valid number of 2 clusters, these two groups are better separated than with the original data set.

When taking into account the ground truth labeling, in other words, real class labels, it is obvious that the cluster labels obtained with original data set have almost nothing in common with the real class labels. External evaluation metrics obtained with the three algorithms indicate, that the labelings with reference to the real class labels are close to random. When using the combination data set with new features included, external evaluation metrics indicate slightly better fit between ground truth labels and the cluster labelings. The scores remain relatively low, since there are 10 different classes, one of which is a class of unlabeled examples and one of which covers more than 81 % of the examples, in contrast to three classes only containing one or two examples. Clustering algorithms do not usually perform well with such unbalanced data sets and thus, these results are not very surprising. However, using the combination data set, it is possible to separate almost all class 2 examples from other classes, and the area of mostly class 1 samples from others. It is also possible to separate the area which contains most of class 9 and 10 samples from the other classes, which are promising results for further research.

Research question 2: *Do distinct learning algorithms perform similarly on the classification task?*

In binary classification, all classes included, the two algorithms used for classification perform remarkably similarly. Random Forest Classifier provides at highest 2.3 % better accuracy scores than SVC on the data set F2, but otherwise the accuracy scores are almost equal for both algorithms. Furthermore, cross-validation scores are almost equal with both algorithms and comparing on both data sets, the biggest difference being 0.5 % on the data set F1. When class 10 is not included in binary classification, Random Forest Classifier provides at highest 1.0 % better accuracy scores comparing on the new data set F2, whereas on data set F1, SVC provides at highest 0.6 % better accuracy scores. Again the cross-validation scores remain almost equal for both algorithms, the biggest difference being approximately 0.4 % on both data sets.

For multiclass case, Random Forest Classifier seems to perform slightly better, providing approximately 1.2 % better accuracy scores, comparing on the new data set F2. The cross-validation scores compared on both data sets are also slightly better (2.2 % at highest) with Random Forest. Random Forests are invariant under scaling and claimed to be robust to irrelevant features. Random Forests also work well with large, even slightly unbalanced data sets. Hence, it is not surprising that

accuracy scores are slightly higher in most cases using Random Forest. More interesting is that the cross-validation scores are relatively similar in all the binary cases for both algorithms, but higher in the multiclass case using Random Forest. This might be due to the fact, that Random Forests are claimed to ease the overfitting tendency of traditional decision trees. Another interesting finding, and one possible explanation, is that Random Forests seem to perform slightly better in differentiating between the smaller classes. This makes a huge difference in the context of this particular data set, even though there are not such a big differences in the accuracy scores.

Research question 3: *Are PFH:s useful in creating new features for classification and clustering tasks, and do these features improve the classification and clustering results?*

The new features obtained by computing PFH:s do not improve classification performance significantly. When all classes are included in the binary classification, accuracy scores are actually lower than on the data set F1, the difference being 2.4 % at highest. However, the cross-validation scores are slightly improving on data set F2, the difference being 0.5 % at highest. When class 10 is not included in the binary classification, 0.2 % better accuracies and 0.4 % better cross-validation scores are obtained with Random Forest. With SVC, accuracy score is approximately 1.4 % lower and the cross-validation score 0.4 % lower on the data set F2. In the multiclass case, Random Forest performs almost equally on both data sets, the differences in accuracy and cross-validation scores being less than 0.1 %. With SVC, the accuracy on new data set is approximately 0.6 % lower than on the original data set. However, the cross-validation scores are again almost equal with both data sets. To conclude, the overall performance of the two algorithms in all approaches is very similar on both data sets. Hence, including PFH-based features does not seem to affect the classification performance significantly.

Evaluating the differences in clustering results is not as straight forward as in classification. The concept of good clustering is not well defined, and it mostly depends on the application. In the context of this data, it seems that the PFH:s are improving the clustering results in the sense of fitting better to the real class labels. In the sense of finding well separated groups in the data, the original data set is providing more distinct sets of points. However, with the reference to the real labels, the structure found in the original data set is almost random. The examples belonging to different classes are not located in the same clusters, but spread across them. This makes the clustering results on the new data set more desirable in the context of this thesis, because the internal structure corresponds better to the ground truth labeling. On the new data set, clustering algorithms are able create such clusters that have something in common with the real labels. In particular, the

algorithms are able to distinguish between examples belonging to class 2 and other classes. It is also possible to create three to four such clusters that include most of class 1 examples. Furthermore, the algorithms are able to create such cluster that includes most of the examples belonging to class 9 and 10. To conclude, in the context of this particular data set, PFH-based features are improving the clustering results in the sense of fitting better to the real class labels, but not necessarily in the sense of internal structure.

These results are promising for further research. Getting labeled data is hard and time-consuming, and requires co-operation between the manufacturing organization and the customer organizations. Hence, if it is possible to create such a clustering that has something in common with the real class labels and can divide the instances in meaningful groups, it is also possible to conduct research on unlabeled data, which is significantly easier to gather. The results in this thesis state that creating such clusters is indeed possible.

In order to get better classification results, it would be beneficial to gather more examples from the smaller classes. One possible extension is to conduct binary classification between two classes, for example 2 vs 4, 2 vs 6, and 4 vs 6, in order to get more insight about the differences and separating conditions between these classes, as well as feature importance for these classifiers. This, in turn, might be helpful in creating and choosing such features that would result in better classification performance in the multiclass case. The results in this thesis show, that it is possible already to separate between all the different classes, but the rareness of defects belonging to the smaller classes limits the possibility to build such model that would accurately separate them in real-life situation. As suggested before, one possible extension would be also the two-step approach, where two classifiers are used sequentially: First binary classification in order to distinguish between class 1 and others, second multiclass classification to distinguish between all the remaining classes. The results of this thesis already confirm that for both binary and multiclass case, quite accurate results can be obtained. It would be interesting to see how well these two classifiers can perform when combined.

Even though the results indicate, that PFH descriptors are not very useful in the classification task, this does not necessarily mean that the PFH:s can not capture the properties of defect shapes. It only indicates, that using PFHs via cumulative sums and as features might not be the best approach. One interesting possibility for further research would be to use PFH descriptors in a more traditional manner, by trying to learn a model for each class via regression, and then use these models in shape recognition.

REFERENCES

[1] Breiman, L. *Random Forests*, Machine Learning, 45(1), 2001, pp. 5-32

[2] Campello, R. J. G. B., Moulavi, D., Sander, J. *Density-Based Clustering Based on Hierarchical Density Estimates*, Lecture Notes in Computer Science, Vol. 7819. Springer, 2013

[3] Chih-Chung C., Chih-Jen L., *LIBSVM: a library for support vector machines*, ACM Transactions on Intelligent Systems and Technology, 2011

[4] Döring, C., Eichhorn, A., Wang, X., Kruse, R. *Improved Classification of Surface Defects for Quality Control of Car Body Panels*, 2006 IEEE International Conference on Fuzzy Systems, 2006, pp. 1476-1481

[5] Estivill-Castro, V. *Why So Many Clustering Algorithms: A Position Paper*, ACM SIGKDD Explorations Newsletter, Vol. 4, 2002

[6] Frey, B. J., Dueck, D. *Clustering by Passing Messages Between Data Points*, Science, Vol. 315, 2007

[7] Hastie, T., Tibshirani, R., Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Second Edition, Springer, 2008

[8] Hotelling, H. *Analysis of a Complex of Statistical Variables into Principal Components*, Journal of Educational Psychology, 24(6 and 7), 1993

[9] Hubert, L., Arabie, P. *Comparing partitions*, Journal of Classification, Vol. 2, no. 1, 1985, pp. 193-218

[10] Izenman, A. J. *Modern Multivariate Statistical Techniques, Regression, Classification, and Manifold Learning*, Springer, 2013

[11] Jia, H., Murphey, Y. L., Shi, J., Chang, T-S. *An intelligent real-time vision system for surface defect detection*, Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004, Vol. 3, 2004, pp. 239-242

[12] Jones, B., Aoun, M. *Learning 3D Point Coud Histograms*, CS229 Machine Learning Project, 2009

[13] Kamani, P., Noursadeghi, E., Afshar, A., Towhidkhah, F. *Automatic Paint Defect Detection and Classification of Car Body*, 7th Iranian Conference on Machine Vision and Image Processing, 2011, pp. 1-6

[14] Karbacher, S., Babst, J., Häusler, G., Laboureux, X. *Visualization and Detection of Small Defects on Car-Bodies*, 1999

[15] Knauer, M. C., Kaminski, J., Häusler, G. *Phase Measuring Deflectometry: a new approach to measure specular free-form surfaces*, Institute of Optics, Information and Photonics, University of Erlangen-Nuremberg, 2004

[16] Lichman, M. *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2013

[17] Lilienblum, T., Albrecht, P., Calow R., Michaelis, B. *Dent detection in car bodies*, Proceedings 15th International Conference on Pattern Recognition. ICPR 2000, Vol. 4, 2000, pp. 775-778

[18] Louridas, P., Ebert, C. *Machine Learning*, IEEE Software, Vol. 33, 2016

[19] McInnes, L., Healy, J., Astels, S. *hdbscan: Hierarchical density based clustering In: Journal of Open Source Software*, The Open Journal, Vol. 2, no. 11, 2017

[20] Michie, D., Spiegelhalter, D. J., Taylor, C. C. *Machine Learning, Neural and Statistical Classification*, Overseas Press, 1994

[21] Moeslund, T. *Introduction to Video and Image Processing: Building Real Systems and Applications*, Springer London, 2012, pp. 103-113

[22] Mohri, M., Rostamizadeh, A., Talwakar, A. *Foundations of Machine Learning*, The MIT Press, 2012

[23] Pearson, K. *On Lines and Planes of Closest Fit to Systems of Points in Space*, Philosophical Magazine, Series 6, 2(11), 1901, pp. 559-572

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

[25] Rosenberg, A., Hirschberg, J. *V-Measure: A conditional entropy-based external cluster evaluation measure*, Columbia University, 2007

[26] Rousseeuw, P. J. *Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis*, Computational and Applied Mathematics 20, 1987, pp. 53-65

[27] Russell, S., Norvig, P. *Artificial Intelligence, A Modern Approach*, Third edition, Pearson, 2010

[28] Rusu, R. B *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*, Künstliche Intelligenz (KI), Vol. 24, no. 4, 2010, pp. 345-348

[29] Rusu, R. B., Cousins, S. *3D is here: Point Cloud Library (PCL)*, IEEE International Conference on Robotics and Automation (ICRA), 2011

[30] Rusu, R. B., Marton, Z. C., Blodow, N., Beetz, M. *Learning Informative Point Classes for the Acquisition of Object Model Maps*, Proceedings of the 10th International Conference on Control, Au- tomation, Robotics and Vision (ICARCV), 2008

[31] Rusu, R. B., Marton, Z. C., Blodow, N., Beetz, M. *Persistent Point Feature Histograms for 3D Point Clouds* In Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS-10), 2008

[32] Tan, P.-N., Steinbach M., Kumar, V. *Introduction to Data Mining*, Pearson, 2006

[33] Tipping, M. E., Bishop, C. M. *Mixtures of Probabilistic Principal Component Analysers*, Neural Computation 11(2), MIT Press, 1999, pp 443-482

[34] van der Maaten, L. *Visualizing Data using t-SNE*, Journal of Machine Learning Research 9, 2008, pp. 2579-2605

[35] Vinh, N. X., Epps, J., Bailey, J. *Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary?*, Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009

[36] Wahl, E., Hillenbrand, U., Hirzinger, G. *Surflet-Pair-Relation Histograms: A Statistical 3D-Shape Representation for Rapid Classification*, IEEE Computer Society Press, 2003, pp 474-481

A. SURFACE PLOTS OF MULTICLASS CLASSIFICATION RESULTS

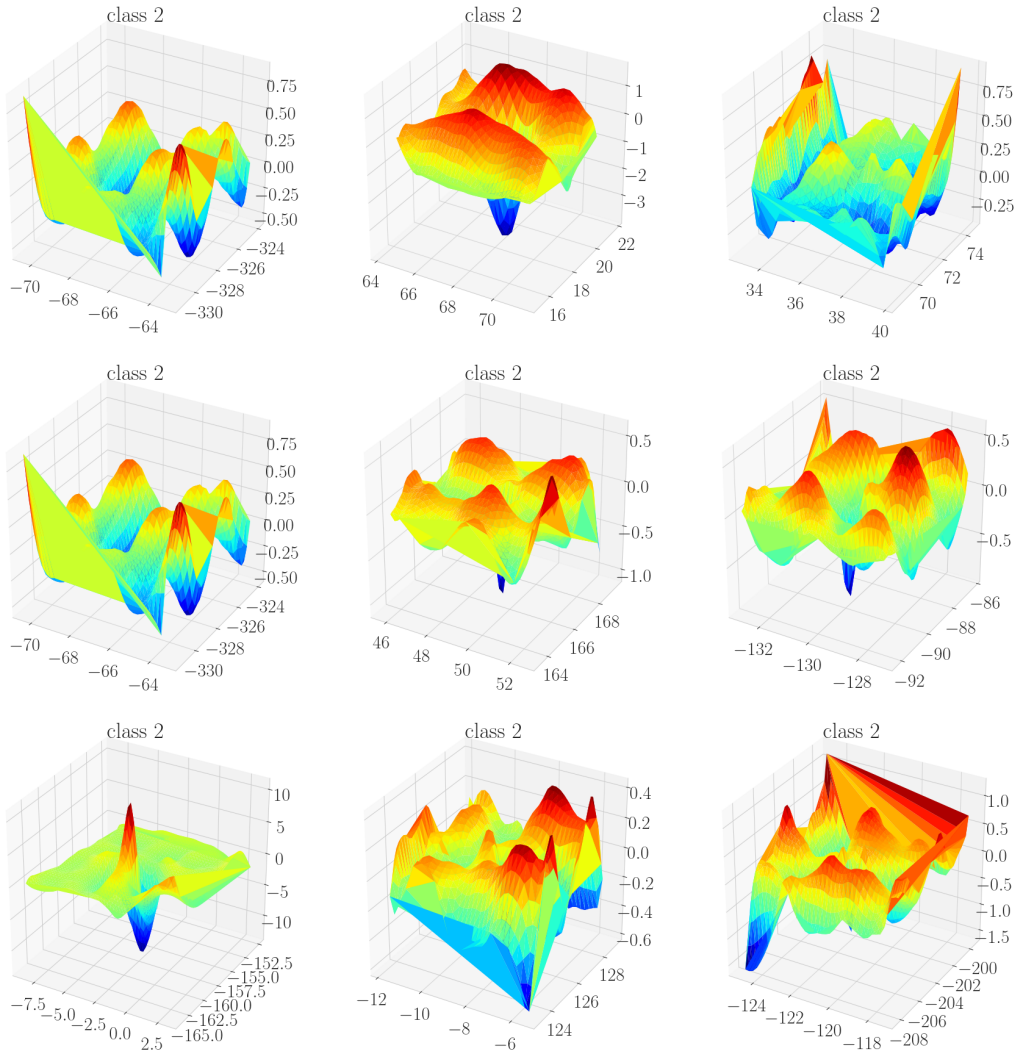

Figure A.1: Class 2 examples which were incorrectly classified as class 1 or class 0 by Random Forest Classifier in the multiclass task. All of these incorrectly classified examples have some "bumps" above the surface and not just crater going below the surface.

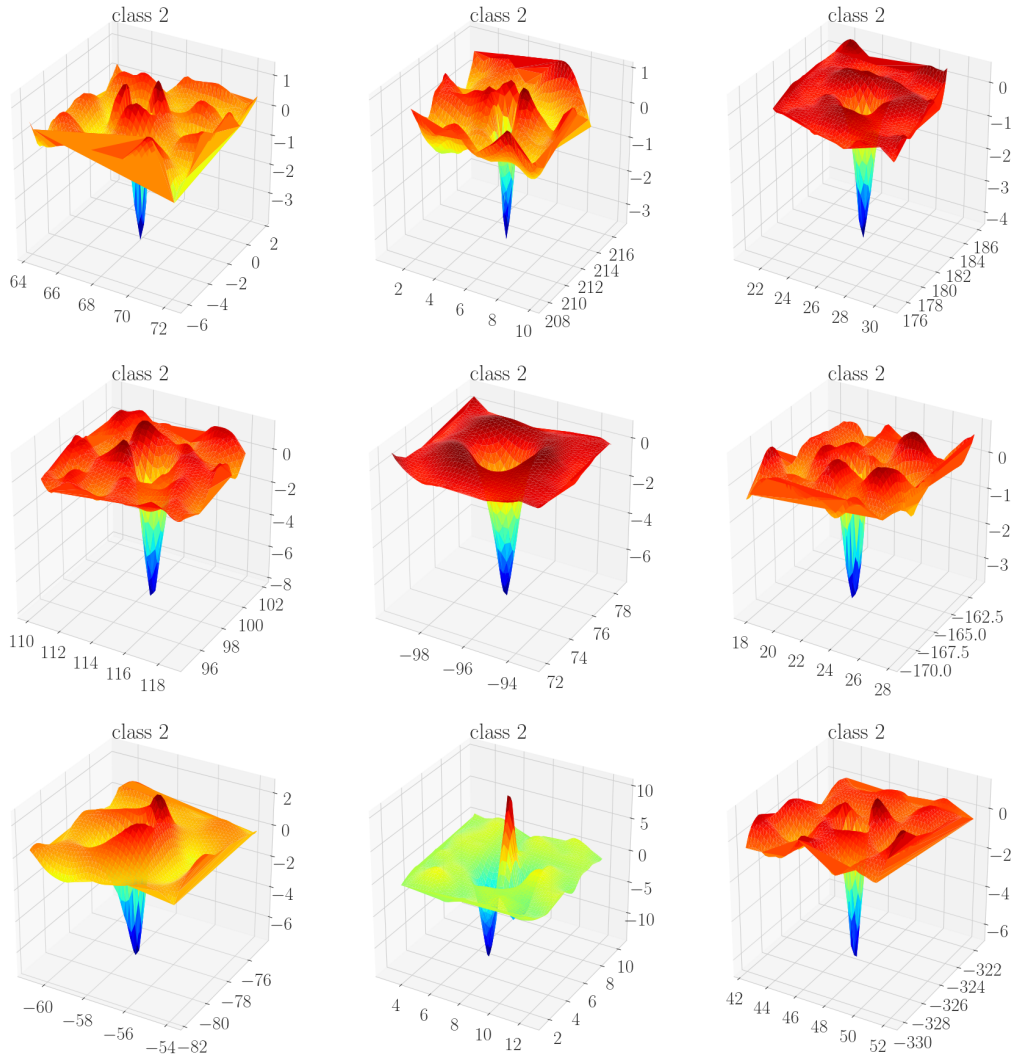

Figure A.2: Class 2 examples which were correctly classified by Random Forest classifier in the multiclass task. Most of correctly classified examples are clearly craters going below the surface, with only little texture above the surface.

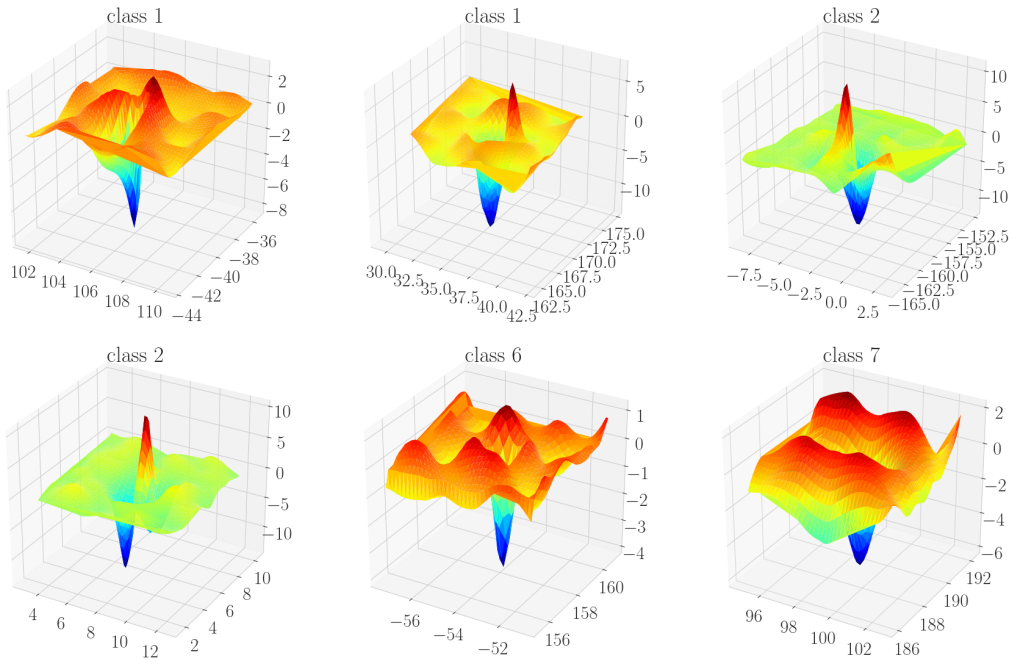

Figure A.3: Examples belonging to different classes. Each of these defects are partly going below the surface and partly above the surface. This demonstrates the challenge in distinguishing between the different classes.

B. FITTING THE GAUSSIAN FUNCTION

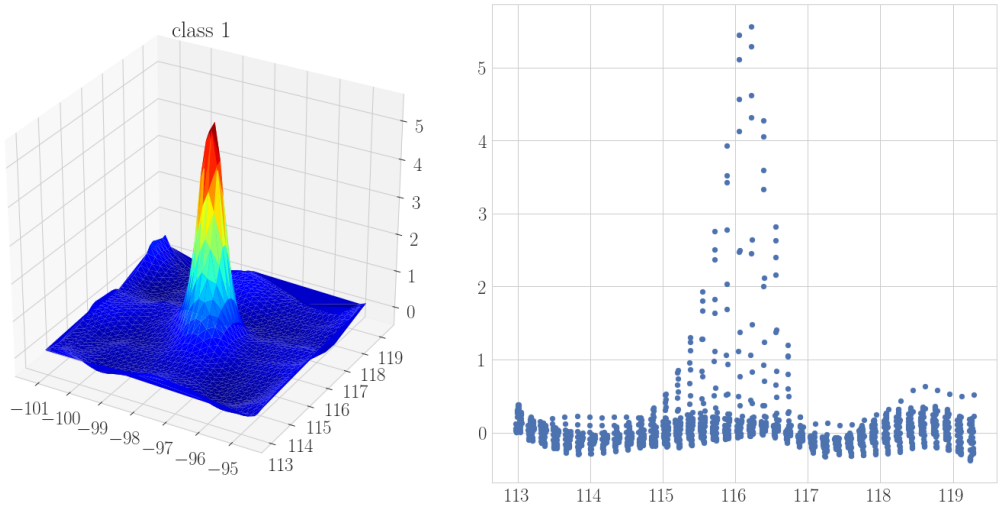

Figure B.1: 3D point cloud and corresponding 2D plot (z-axis) of class 1 example.

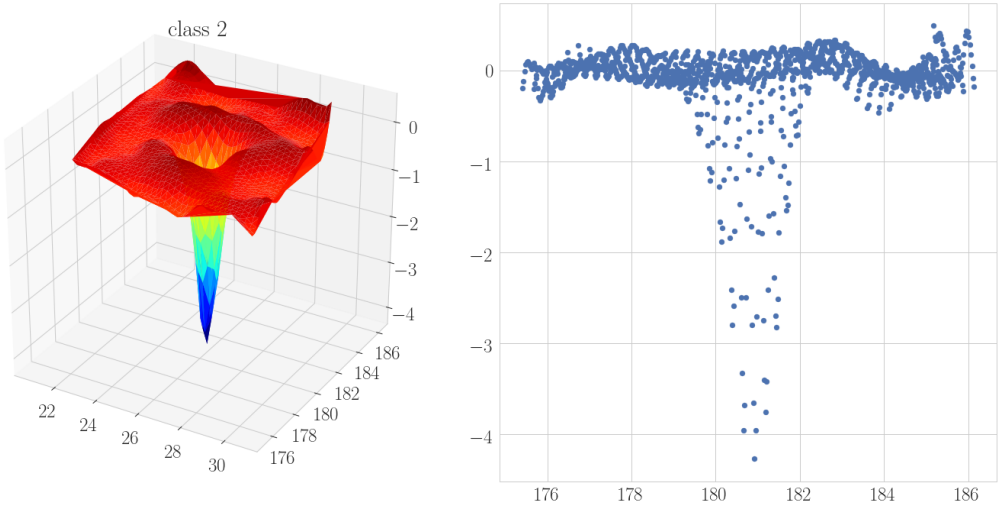

Figure B.2: 3D point cloud and corresponding 2D plot (z-axis) of class 2 example.

From Figure B.1 and Figure B.2, it can be seen that considering only z-axis of 3D point clouds, there is some resemblance with gaussian curves. Hence, new features were obtained by fitting gaussian function:

$$f(x,y) = A{\cdot}\exp\left(-\frac{1}{2(1-\rho^2)}\cdot\left(\frac{(x-\mu_x)^2}{\sigma_x^2} - \frac{2\rho(x-\mu_x)\cdot(y-\mu_y)}{\sigma_x\sigma_y} + \frac{(y-\mu_y)^2}{\sigma_y^2}\right)\right)+B,$$

from where it is possible to compute the following features:

Feature	**Formula**	**Explanation**
amplitude_z_um	A	Amplitude of Gaussian function (μm)
offset_z_um	B	Vertical offset of the Gaussian function (μm)
center_x_mm	μ_x	x-coordinate of the mean value (mm)
center_y_mm	μ_y	y-coordinate of the mean value (mm)
sigma_major_axis_mm	$\sigma_x^{'}$	Variance along the main axis (mm)
sigma_minor_axis_mm	$\sigma_y^{'}$	Variance along the smaller axis (mm)
orientation_angle_rad	ϑ	Angle of rotation in radians (computed from ρ, σ_x, and σ_y)
fitting_error_um2	$\sum_i dist(P_i - f(x,y))^2$	Fitting error (μm^2)
fitting_rms_um	$\sqrt{\frac{\text{fitting\_error\_um2}}{\text{number\_points}}}$	Mean square error (μm)
volume_gauss_mm3	$V = 2\pi A\sigma_x^{'}\sigma_y^{'}$	Volume under the fitted Gaussian function (mm^3)
volume_positive_mm3	$\sum_i z_i$ for $z_i > 0$	Volume over XY-plane (mm^3)
volume_negative_mm3	$\sum_i z_i$ for $z_i < 0$	Volume under XY-plane (mm^3)
volume_total_mm3	volume_positive_mm3 + volume_negative_mm3	Total volume (mm^3)