OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Exploiting Background Knowledge on Evolving Objects to Identify Relevant Dimensions for Classification

DISSERTATION

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von M.Sc. Tommy Hielscher

geb. am 25.04.1987 in Magdeburg

Gutachterinnen/Gutachter

Prof. Dr. Myra Spiliopoulou
Prof. Dr. Bernhard Preim
Prof. Dr. Panagiotis Papapetrou

Magdeburg, den 26.03.2019

Otto von Guericke University Magdeburg

Faculty of Computer Science

Knowledge Management and Discovery

Dissertationsschrift

Zur Erlangung des akademischen Grades: Dr.-Ing

Exploiting Background Knowledge on Evolving Objects to Identify Relevant Dimensions for Classification

Author:

Tommy Hielscher

26. März 2019

First Supervisor: Prof. Myra Spiliopoulou
Second Supervisor: Prof. Bernhard Preim
Third Supervisor: Prof. Panagiotis Papapetrou

Hielscher, Tommy:
Exploiting Background Knowledge on Evolving Objects to Identify Relevant Dimensions for Classification
Dissertationsschrift, Otto von Guericke University Magdeburg, 2019.

Abstract

Classification models are widely used in a plethora of different applications to automatically assign objects into one of several pre-defined categories. Typically, objects are represented by multi-dimensional feature vectors and classification models are induced by learning associations between features and the class variable from a set of objects where the class is known. In reality, objects can be complex, change over time and may come with additional background and meta-information. Standard classification algorithms and workflows often do not fully use certain types of information about objects, or do not use specific kinds of background knowledge at all, to detect the dimensions that are actually relevant w.r.t. the target concept. This can lead to the inclusion of irrelevant feature into models, possibly leading to sub-par performance. Therefore, identifying the dimensions that are relevant w.r.t. the target concept is essential in many domains to produce quality classification models and provide experts the possibility to deepen domain understanding. We argue that different type of background knowledge can be utilized to overcome this challenge.

In this thesis, we propose a framework that exploits three kinds of background knowledge to identify relevant implicit and explicit dimensions of evolving objects. The first component exploits ground truth (class-label data) about the target concept and historical records to extract the dimensionality that is implicitly given by an object's evolution and codify this knowledge into new features. The framework detects clusters of similar concept related patterns in the historical sequence of the objects feature-values and abstracts them into nominal features. It further analyzes and codifies statistics from the evolution of single objects and groups of similar objects.

The second framework component presents our constraint-based subspace selection algorithm DRESS, which detects a set of relevant features from the set of explicit object features. For this, DRESS utilizes background knowledge in the form of similarity constraints between objects to rank feature subspaces according to their relevance regarding the target concept. DRESS prefers feature subspaces where objects under similarity constraints exhibit small distances and are members of the same cluster, and where objects under dissimilarity constraints exhibit high distances and are members of different clusters.

Additionally, our framework contains a component where the results of DRESS can be validated on independent datasets. First, for the validation, both datasets are made comparable by only retaining the instances with similar distribution in selected covariates. Then, clusters found by DRESS that were used for the decision on relevant features are transferred on the validation dataset. Finally, the validation component accesses whether the transferred clusters exhibit similar class distribution and size compared to the original ones.

Our framework is evaluated on real-world epidemiological study data. We show the impact on classifier performance of the several framework components for different base classifiers: It is investigated to what extend the proposed methods enhance base classifier performance when deriving new features from an object's (or groups of objects) evolution. We evaluate the classifier performance when using the feature subspaces derived by DRESS in comparison to the feature sets proposed by traditional feature selection approaches. Further, we investigate whether literature supports findings of our framework w.r.t. the identified associations between features and target concept.

Results show that the features derived by the first framework component enhance the majority of investigated base classifiers and are used by all variants that achieved best overall performance on their respective dataset. Regarding the selection of feature subsets, DRESS is the most stable feature selection algorithm in our evaluation as it identifies feature subsets that improve the performance of all base classifiers and that produce consistently competitive performance across all evaluated datasets, classification algorithms and training data subsets.

Zusammenfassung

Klassifikationsmodelle werden in vielen unterschiedlichen Szenarien verwendet, um automatisch Objekte zu einer von mehreren vordefinierten Kategorien zuzuordnen. Typischerweise werden solche Objekte durch multi-dimensionale Feature-Vektoren beschrieben, wobei die Klassifikationsmodelle durch Erlernen der Zusammenhänge zwischen den Features und der Klassenvariable, von Objekten mit bekannter Klassenzugehörigkeit, induziert werden. In der Realität können Objekte komplex, sich verändernd über die Zeit und möglicherweise mit zusätzlichen Hintergrund- und Metainformationen assoziiert sein. Traditionelle Klassifikationsmethoden und -abläufe verwenden häufig bestimmte Arten von verfügbaren Objektinformationen nicht, oder nicht vollständig, um die Dimensionen zu ermitteln die eine Relevanz bzgl. des Zielkonzepts aufweisen. Dies kann zur Inklusion von irrelevanten Features in Klassifikationsmodellen führen, und damit zu unterdurchschnittlicher Klassifikationsperformanz. Zur Erstellung von qualitativen Klassifikationsmodellen und der Vertiefung des Domänenwissens von Experten ist es daher essentiell die Dimensionen zu identifizieren welche relevant für das Zielkonzept sind. Wir argumentieren, dass unterschiedlichen Arten von Hintergrundwissen über evolvierende Objekte genutzt werden können, um dieses Problem zu lösen.

In dieser Dissertationsschrift schlagen wir ein Framework vor, welches drei verschiedene Arten von Hintergrundwissen nutzt um relevante implizite und explizite Dimensionen von evolvierenden Objekten zu identifizieren. Die erste Framework-Komponente verwendet Ground Truth

(Klassen-Label Daten) über das Zielkonzept und historische Datensätze der Objekte, um die implizite Dimensionalität, gegeben durch die Objektevolution, zu explizieren und dieses Wissen anschließend in neuen Features zu kodifizieren. Das Framework erkennt Cluster von ähnlichen, konzeptrelevanten Mustern in der Historie der Feature-Werte von Objekten und abstrahiert diese durch nominale Features. Weiterhin werden Evolutionsstatistiken von einzelnen Objekten und Objektgruppen analysiert und kodifiziert.

Die zweite Framework-Komponente präsentiert unseren eigens entwickelten Constraint-basierten Subspace Selection Algorithmus DRESS, welcher eine Untermenge von relevanten Features von der Menge aller explizit gegeben Objekt-Features selektiert. Hierfür verwendet DRESS Hintergrundwissen über die Ähnlichkeit zwischen verschiedenen Objekten, um Feature-Räume entsprechend ihrer Zielkonzeptrelevanz zu bewerten. DRESS bevorzugt dabei Feature-Unterräume in denen Objekte unter Ähnlichkeits-Constraints kleine Distanzen aufweisen und identischen Clustern zugeordnet sind, und in denen Objekte unter Unähnlichkeits-Constraints hohe Distanzen zueinander aufweisen und unterschiedlichen Clustern zugeordnet sind.

Zusätzlich beinhaltet unser Framework eine Komponente zur Validierung der Ergebnisse von DRESS auf unabhängigen Datensätzen. Hierzu werden der ursprüngliche Datensatz und der Validierungsdatensatz auf die Objekte reduziert, welche eine ähnliche Verteilung bzgl. vordefinierter Kovariaten aufweisen. Danach werden die Cluster, auf deren Grundlage DRESS die Relevanz von Feature-Unterräumen bewertet hat, auf den Validierungsdatensatz übertragen. Schlussendlich evaluiert die Validierungskomponente ob die übertragenden Cluster eine ähnliche Klassenverteilung und Größe, verglichen mit den ursprünglichen Clustern, aufweisen.

Unser Framework wird auf epidemiologischen Studiendaten evaluiert. Wir zeigen den Einfluss der verschiedenen Framework-Komponenten auf die Klassifikationsperformanz unterschiedlicher Basisklassifikatoren: Es wird untersucht inwieweit die vorgeschlagenen Methoden die Performanz der Basisklassifikatoren verbessern, wenn neue Features aus der

Evolution einzelner Objekte, oder ganzer Objektgruppen, abgeleitet werden. Wir evaluieren die Klassifikatorperformanz bei Nutzung der von DRESS identifizierten Feature-Räume im Vergleich zu den Feature-Mengen die durch traditionelle Feature-Selection Algorithmen gefunden werden. Weiterhin wird untersucht ob die Literatur die durch unser Framework erzielten Ergebnisse, bzgl. der identifizierten Assoziationen zwischen Features und Zielkonzept, unterstützt.

Die Ergebnisse zeigen dass Features welche durch die erste Framework-Komponente abgeleitet wurden, eine Verbesserung der Performanz eines Großteils der untersuchten Basisklassifikatoren erzielen, und dass diese Features von allen Varianten verwendet werden welche die insgesamt beste Performanz auf den unterschiedlichen Datensätzen erreichen. Zudem konnte DRESS im Hinblick auf die Selektion relevanter Feature-Unterräume die stabilsten Ergebnisse erzielen: DRESS identifiziert Feature-Räume welche die Performanz der Basisklassifikatoren verbessert und welche konsistent konkurrenzfähige Performanz über alle evaluierten Datensätze, Klassifikationsalgorithmen und Trainingsdaten-Mengen erzielen.

Contents

Publications

List of relevant publications w.r.t. the thesis topic to which the author contributed:

T. Hielscher, M. Spiliopoulou, H. Völzke, and J. P. Kühn, ”Using participant similarity for the classification of epidemiological data on hepatic steatosis,” in *Proc. of the 27th IEEE Int. Symposium on Computer-Based Medical Systems (CBMS14).* Mount Sinai, NY: IEEE, May 2014, pp. 1-7.

Short summary: In this work, it is studied how similarity measures for complex objects contribute to class separation for a multifactorial disorder. A workflow for data preparation, partitioning and classification of cohort participants for the disorder ”hepatic steatosis” is presented. Authors report on findings w.r.t. classifier performance and important features related to the disorder.

T. Hielscher, M. Spiliopoulou, H. Völzke, and J.-P. Kühn, ”Mining Longitudinal Epidemiological Data to Understand a Reversible Disorder”, in Advances in Intelligent Data Analysis XIII: 13th International Symposium, IDA 2014,Leuven, Belgium, October 30 - November 1, 2014. Springer International Publishing, 2014, pp. 120-130. [Online].
Available: http://dx.doi.org/10.1007/978-3-319-12571-8 11

Short summary: In this work, it is investigated how sequences of historical records of complex objects can improve class separation. To this means, a method that derives feature relevant to the target concept from sequences of historical object records is developed and presented. It is shown that the method derives features that improve predictive quality and that factors that contribute inadequately to class separation become more predictive.

U. Niemann, T. Hielscher, M. Spiliopoulou, H. Völzke, and J.-P. Kühn, ”Can we classify the participants of a longitudinal epidemiological study from their previous evolution?” in *Proc. of the 28th IEEE Int. Symposium on Computer-Based Medical Systems (CBMS15).* São Carlos and Ribeirão Preto, Brazil: IEEE, June 2015, pp. 121-126. [Online]. Available: http://dx.doi.org/10.1109/CBMS.2015.12

Short summary: In this work, a method for the codification of features based on the evolution of single objects and groups of objects is presented. The method encompasses steps for tracing the evolution of complex objects and for subsequently using derived features in classification. It is shown that the presented approach improves class seperation and that the change (over time) in the values of features is predictive in a real-world medical domain.

T. Hielscher, M. Spiliopoulou, H. Völzke, and J.-P. Kühn, ”Identifying relevant features for a multi-factorial disorder with constraint-based subspace clustering” in C*BMS2016: Proceed- ings of the 29th IEEE Symposium on Computer-Based Medical Systems.* Dublin, Ireland: IEEE, 2016, pp. 207-212.

Short summary: In this work, an algorithm which uses constraints on the similarity between examples to discover feature subspaces that describe a target concept is presented. The algorithm exploits the density of clusters and the distance-behavior between constrained objects to evaluate the quality of a feature set without requiring explicit class label information about the target variable.

S. Alemzadeh, T. Hielscher, U. Niemann, L. Cibulski, T. Ittermann, H. Völzke, M. Spiliopoulou, and B. Preim. ”Subpopulation Discovery and Validation in Epidemiological Data” in *EuroVis 2017: Proceedings of the 8th Int. EuroVis workshop on Visual Analytics*, Barcelona, Spain: Eurographics Association, 2017, pp. 43-47.

Short summary: In this work, an interactive coordinated multiple view system ”Visual Analytics framework for SubpopulAtion Discovery and Validation In Epidemiological Data” (S-ADVIsED) is presented. The proposed system enables medical experts the exploration and validation of findings derived from subspace clusterings.

T. Hielscher, H. Völzke, P. Papapetrou, and M. Spiliopoulou, "Discovering, selecting and exploiting feature sequence records of study participants for the classification of epidemiological data on hepatic steatosis" in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, ser. SAC '18. New York, NY, USA: ACM, 2018, pp. 6-13. [Online]. Available: http://doi.acm.org/10.1145/3167132.3167162

Short summary: In this work, a framework for Discovery, Selection and Exploitation (DiSelEx) of longitudinal epidemiological data is presented. The framework aims to identify informative patterns among historical sequences of the complex objects under study. Here, sequence clustering and supervised learning for the identification of sequence groups that contribute to class separation is employed.

T. Hielscher, U. Niemann, B. Preim, H. Völzke, T. Ittermann, and M. Spiliopoulou, "A framework for expert-driven subpopulation discovery and evaluation using subspace clustering for epidemiological data" *Expert Systems with Applications*, vol. 113, pp. 147-160, 2018. [Online]. Available: http://www. sciencedirect.com/science/article/pii/S0957417418304202

Short summary: In this work, an intelligent system that assists domain experts in analyzing the data of complex objects is proposed. The system supports the expert in identifying relevant features for a target concept derived from subspace clusters with distinct class distributions. Further, a module for the validation of findings (concerning variables and subspace clusters) on independent datasets is presented.

List of Figures

List of Tables

List of Algorithms

List of Acronyms

CFS	Corrlation-based Feature Selection
DD	Discovery Dataset
DRESS	Discovering Example-constrained SubSpaces
GBT	Gradient Boosted Trees
GR	Gain Ratio
HEOM	Heterogeneous Euclidean Overlap Metric
IG	Information Gain
kNN	k Nearest Neighbor
ML	Must-Link
MLP	Multilayer Perceptron
MONICA	Multinational MONItoring of trends and determinants in CArdiovascular disease
NB	Naive Bayes
NL	Not-Link
RCDD	Relative Class Distribution Difference
RF	Random Forest
RQ	Research Question
RSD	Relative Size Difference
SVM	Support Vector Machine
SHIP	Study of Health in Pomerania
VD	Validation Dataset

List of Symbols

s	sequence-example
t	time-point
x	object/instance
C	cluster
D, D_{te}, D_{tr}	complete dataset, test set, training set
F	set of static features
G	set of time-series features
S	sequence-set
Y	set of class-labels
π_F	projection on feature space F
$\mathcal{C}, \mathcal{CL}$	set of clusters, set of clusterings
$\mathcal{C}_{f,D}$	a clustering (set of clusters) on the projection of D on f
$\mathcal{D}_{tr}$	a super set of training instance partitions
$\mathcal{ML}$	set of Must-Link constraints
$\mathcal{NL}$	set of Not-Link constraints

1. Introduction

Classification is the task of assigning objects to one of several predefined categories and is widely used in many different domains. Objects can be any entities such as patients, e-mails or galaxies. Classification models support decision making in their respective domain, for example, supporting medical experts in the diagnosis of a disease [1–3], helping crime scene investigators to determine the type of a piece of evidence [4–6] or detecting unauthorized credit card usage [7–9]. Data mining algorithms can help to automatically build such classification models for different domains. Here, models are derived from sets of objects where the category (class) assignment is known. The data that is used for model learning is highly influential on the resultant model quality. Objects are often described by a plethora of different features which are not all relevant but may also contain irrelevant information regarding the class-variable. Therefore, classification models do not always make the right decision for objects up for classification. Additionally, domain experts are interested in the specific dimensions and data that led to the classification performance at hand. Hence, experts might still want to analyze the specific features that could be used for their respective classification problem while they are not even able, or care, to create classification models (for instance due to lack of ground-truth). Especially when the task is to gain a deeper understanding of a specific domain, then the features that can be used for a classification problem are more important than the classification model itself. For example, medical experts might be interested to investigate the shared characteristics of subpopulations that can be used to classify a medical outcome [10] or investors might base their decisions on the indicators that performed best for stock market prediction [11].

Further, the common assumption that objects are simply described by single, multi-dimensional feature vectors does not hold true in both of the previous examples: stock market indicators and characteristics of individuals change over time. This holds true for many real-world domains where objects evolve over time and are repeatedly observed. Such evolution can implicitly contain additional dimensions that are relevant w.r.t. the target concept and the domain expert. Many traditional approaches struggle to derive, identify and present relevant dimensions from evolving objects regarding a classification problem, by either being not applicable on such objects (e.g. [12], [13], [10]) or by not making the relevant information given by the sequences of historical records explicit (e.g. [14], [15, 16]). This in turn leads to sub-par classification models and/or domain understanding. Hence, providing domain experts with the capabilities to automatically derive dimensions from the evolution of objects, identify the relevant dimensions and access their associations w.r.t. the target concept and between each other is a key challenge here. We argue that this challenge can be overcome when different types of background knowledge about the relationship between objects or their relevance w.r.t. the target concept, given by domain experts or literature, are exploited.

In the next section we motivate the need to exploit different types of background knowledge for evolving objects to identify relevant dimensions for classification tasks.

1.1 Motivation to Exploit Different Types of Background Knowledge of Evolving Objects

Identifying relevant dimensions that are associated with a target concept poses numerous challenges to data mining algorithms. Some of those challenges are the application of traditional methods to non-static objects, codifying the implicit knowledge inherent to the objects evolution, identifying relevant dimensions when class-label data is unavailable or scarce, finding the dimensions that are actually relevant to both the expert and target concept etc. We address these issues by proposing a framework for the identification of relevant dimensions which incorporates our methods that utilize three different types of background knowledge:

- Ground-truth in the form of class-labels
- Object evolution
- Object similarity

Ground truth in the form of class-label data is used to access the relevance of dimensions and patterns towards the target concept. Traditionally, methods exclusively exploit ground truth (without incorporating other types of knowledge) to find relevant dimensions for static objects [17]. We propose methods that use ground-truth in conjunction with the remaining types of background knowledge to guide our algorithms and derive relevant features from the evolution of objects.

Object evolution as given by historical records of objects is exploited to find temporal patterns of an object's evolution within temporal features. Methods that utilize this sort of knowledge often require prior abstraction of features into nominal categories (e.g. [18]), manual parameter settings (e.g. [19]) and do not consider to derive meaningful knowledge from the evolution of similar objects [20]. Such abstraction may lead to a loss of information or may be executed without considering the relevancy of the data at hand. Therefore, we propose methods to find univariate patterns in a data-driven way (w.r.t. feature abstraction and manual parameter settings) and describe the multi-variate evolution of whole groups of objects according to pre-defined functions.

Object similarity in the form of similarity constraints is utilized to identify groups of objects that satisfy such constraints in the complete dataset [21]. Because class-label data may be scarce or unavailable, we argue that such constraints can also be exploited to find sets of relevant dimensions by investigating generated clusters w.r.t. these constraints. We propose a method to determine the relevance of feature subspaces without requiring a large number of class-label data with the help of similarity constraints.

Our proposed framework that exploits the aforementioned types of background knowledge focuses on two essential tasks that are related to the identification of relevant dimensions: (1) the identification of relevant dimensions that are explicitly available (via features), and (2) the identification of dimensions that are implicit to the evolution of objects. Because both tasks are strongly associated with each other, they must be analyzed in conjunction: Consider an object with a number of characteristics that change and are observed multiple times during the object's evolution. For example, we may encounter dimensions that are only relevant during specific time-points within the evolution, regarding the classification task at hand. This in turn leads to the necessity of altering the object description to only reflect information that is useful for classification, and disregard the additional irreleveant information. On the other hand, exploiting the object's evolution may lead to the definition of a plethora of new features/patterns that could be used for classification (for example features containing the number of significant changes or features that describe simple statistics of an object's evolution). Naturally, not all of these derived dimensions may contain relevant information w.r.t. the target concept, thus making it necessary to again find the derived dimensions that are actually relevant. This means that the results of (2) must be actually input to (1), which in turn alters the results of (1) (compared to it's results without using (2)). In the following subsections we further motivate both tasks.

1.1.1 Identifying Relevant Explicit Dimensions

Classification models are derived from objects with known class. They model the associations between the features which describe the objects and the different classes. Therefore, the quality of the generated classification model is determined by the choice of objects and the features used for their description. The quality of many classification algorithms is reduced when objects are described by redundant and / or irrelevant features [17, 22, 23]. Those features may induce models with false and random feature-class associations which are only present by chance, or the models may overweight associations that are captured multiple times by redundant features. This sub-task describes the challenge of identifying relevant dimensions from the set of explicitly given dimensions with the help of background knowledge. Here, only the dimensions which are explicitly used to describe an evolving object are investigated. For non-evolving objects and labeled training data, selecting the right features for classification model learning is well studied in literature [24]. However in a setting where objects are described by multiple feature-vectors (one for each observation) and the absence of an adequate number of traditional background knowledge (i.e. labeled training examples / ground truth), these methods do not suffice. In the following we motivate this with two examples.

Example 1 - Evolution-dependent Relevance

Consider the target concept ”current blood-sugar level” and an individual for whom various measurements are executed each hour over several days. While an example feature like ”number of consumed soft drinks in the last hour” is certainly relevant to determine whether the current blood-sugar level is elevated or not, only the subset of recent observations for this feature constitutes this relevancy instead of all available observations. Generally, for any target concept, features may be only relevant during specific points in time during the evolution of the object. Hence, the number of consumed soft drinks two days earlier is not relevant for the current blood sugar level. This translates to the necessity to shift the holistic view on relevance of features towards an observation-based view, dependent on the evolution of the object.

Another challenge comes with the possible absence of an adequate amount of traditional background knowledge (i.e. labeled training examples / ground truth) and is motivated in example 2.

Example 2 - Lack of Traditional Background Knowledge

Traditional methods for the identification of relevant dimensions use (often extensive amounts of) ground truth (class-label data, i.e. [12, 23, 25]) to access the relevancy of dimensions. However, many application domains face the challenge that no class-label data (or only a limited amount) for the evolving objects under investigation is available. For example, labeling of available data might be too expensive or undesired (target concept exists but the granularity of classes that should be defined is uncertain). In such cases, relevant sets of dimensions cannot be found or the relevance of sets is very uncertain. For these special circumstances it is essential to find different ways to access the relevance and identify relevant dimensions.

For both example 1 and example 2 we propose methods within our framework that address these challenges. We show how a traditional approach can be applied on evolving objects by utilizing a pre-processing method that changes the model used for object representation. The problem of scarce ground-truth when identifying relevant sets of dimensions is dealt with by utilizing an object-similarity based subspace clustering algorithm that utilizes either domain expert knowledge or a small number of class labeled data.

1.1.2 Deriving Relevant Knowledge From Object Evolution

Static objects described by a single feature-vector provide only information regarding their difference in the feature-values in comparison to other static objects. On the other hand, evolving objects additionally differ in direction, speed of change, shape etc. of their evolution-trajectories. Here, the individual available feature-values do not exclusively contain the relevance towards the target concept. Instead, the evolution throughout time may implicitly hold important information. It is therefore necessary to analyze not just each individual observation of a feature but their sequential observations and induced patterns. This sub-task deals with the challenge of making such implicit knowledge explicitly available for an object's representation with the help of background knowledge. Background knowledge like similarity of evolving objects and ground-truth w.r.t. the target concept may provide hints on boundaries of important/unimportant sequences of feature-values or evolution statistics.

Finding patterns and deriving statistics from object evolution has been investigated thoroughly in literature (cf, [20, 26]). However, it is often difficult to apply existing methods due to a requirement to abstract continuous variables into categorical variables beforehand (without knowing if the abstraction eliminates relevant knowledge), parameters that are hard to set for domain experts and lack of an incorporation of similar objects when deriving knowledge. The following two examples motivate our approaches on the analysis of object evolution.

Example 3 - Sequential Patterns

In many domains historical records about objects exist. For example, patients that visit the same hospital multiple times where on each occasion an electronic health record (EHR) is captured and the patient is described by the record sequence.

To diagnose a disease or disorder, single feature-values within one EHR may not contain enough information. Instead, patterns for feature-values may contain this vital knowledge. It is possible that the overall direction of the patient's evolution or significant changes in it's evolution signal the onset of a disease or disorder that would be missed if only the explicitly given feature-values are considered. However, such (human-interpretable) patterns are only detected for categorical features. Any continuous feature must be abstracted into categories beforehand. When the target concept is not considered during such abstraction, it can lead to the loss of relevant information because the newly defined categories may share no longer associations with the target concept. Therefore, it is necessary to abstract continuous features that are observed multiple times into arbitrary shaped patterns in a data-driven way (patterns derived from similar sequential values that are frequently observed) while taking the relevancy towards the target concept into account, instead of detaching the abstraction step from the pattern-discovery step.

Only considering sequential patterns is not sufficient. Example 4 motivates the requirement to investigate statistics provided by the evolution of objects.

Example 4 - Evolution Statistics

Consider the target-concept "current fitness level" and an athlete for whom various measurements are executed each day. Statistics like mean and variance for the daily measured feature "duration of high-intensity workout" provide relevant information w.r.t. the athlete's current fitness level. In comparison, single (or even multiple) specific values for the aforementioned feature do not contain this information: an individual is neither fit nor unfit because of single workouts, instead the state of the target variable depends on statistics observed for the feature over time. Additionally, the athlete may also be part of a group of other athletes that share some characteristics, like the same training plan, participation in sports disciplines etc. Here, the behavior of the group as well the relative behavior of individuals (for example, the athlete may drop out of a highly competitive group, into a more casual one) may have associations with the target concept. We argue that analyzing static and evolutionary statistics of groups of objects and the relation of single objects towards those groups (over time), in addition to the analyzes of traditional object-level statistics, may lead to new insights that can improve the performance in the respective classification task.

To avoid the possibility of disregarding important information for classification purposes when dealing with evolving objects we propose methods to codify the knowledge hidden in the feature-value sequences (regarding example 3). We further analyze what information can be derived from similar evolving objects and if the identification of such groups can be beneficial during classification and for domain experts (example 4).

1.2 Research Questions

The described motivation leads to the following core research question (RQ) that is dealt with in this thesis.

Core Research Question:

How to find relevant dimensions of evolving objects for classification tasks with the help of background knowledge?

In this thesis we investigate how different kinds of background knowledge can be used to find relevant dimensions (w.r.t. a concept) of objects which are described by short sequences of high-dimensional vectors. The core research question is covered by our proposed framework and can be decomposed into sub-questions that are associated with the methods contained within the framework. The answer for the core question is derived by answering the sub-questions within this section.

RQ 1: How can we codify the implicit dimensionality of an object's historical records to improve object classification?

In comparison to static objects, the features of evolving objects may be observed several times. Instead, sequential feature-value patterns can emerge and we can observe how objects move throughout the feature space in relation to their peers over time. In RQ 1 it is investigated how the additional information about the record sequence of an object can be used to codify this dimensionality inherent to the object's evolution, and how this information impacts the classification task at hand. To answer this question, we compare classification workflows on evolving objects that use features derived from their evolution (with the help of self-developed methods) to workflows that only use the explicitly given data. Further, to answer RQ 1, more specific sub-research question must be formulated that cater to two different approaches for the codification of an object's evolution.

RQ 1.1: How to identify relevant feature-value sequence patterns of evolving objects for classification while preserving relevant information?

Evolving objects are associated with a sequence of high-dimensional feature-vectors, with each feature-vector describing the object during a specific point in time. Relevant feature-value sequences are sequences which provide information to improve classification of the target variable. As discussed in the motivation, the often detached discretization (for continuous features) or abstraction (for nominal features) step from the discovery of patterns can cause information loss. RQ 1.1 is about the question how such patterns are found in a way that avoids the aforementioned problem. In the context of RQ 1, the sub-question RQ 1.1 represents the univariate pattern-based view on the codification of object evolution. To answer RQ 1.1 we propose a method on the identification of such relevant feature-value sequence patterns in a data-driven way where the abstraction/discretization is part of the pattern-discovery.

RQ 1.2: How to model the evolution of complex objects for classification?

The evolution of objects encompasses a plethora of information that can be modeled and afterwards analyzed regarding their association with the target. From one moment to the next, the absolute and relative position, object groupings etc. may change. RQ 1.2 is about the question how such information can be modeled and utilized for classification tasks. It is answered by providing a method that derives evolutionary descriptors from objects. In the context of RQ 1, the sub-question RQ 1.2 represents a deductive, multivariate group-based view on the codification of object evolution.

RQ 2: How to identify the relevant features of evolving objects by exploiting background knowledge?

This second RQ is concerned with a cornerstone of classification, the representation of the objects that are classified. Typically objects are described by a set of explicitly given features which represent the dimensions of the data space. Classification performance relies on an adequate dimensional representation of objects. Object representations which encompass redundant or irrelevant dimensions can lead to bad classification performance because the model is built on the wrong or on too much of the same information. To avoid that, it is necessary to find sets of dimensions / features that are relevant, regarding the classification task at hand. In this second research question we deal with the problem of how to find relevant dimensions of evolving objects from the set of explicitly given features with the help of a specific kind of background knowledge. To answer this question, we compare classification workflows that use subsets of the complete feature space to workflows that use all given features. If our method identifies relevant feature subsets, the performance of the classification workflows that use these sets should exceed the performance of the baselines. There exists a plethora of research regarding the identification of relevant feature subsets with the help of ground-truth. However, when no or only limited ground-truth exist, traditional feature selection approaches cannot be executed. A different type of background knowledge is given by instance-level constraints. Those constraints express knowledge about the similarity between two objects without necessarily expressing what this similarity constitutes. RQ 2 is about the question how we can find relevant feature sets in scenarios where the typical background knowledge (ground-truth) is scarce or unavailable. To answer RQ 2 we propose a method that utilizes instance-level constraints to find relevant features.

1.3 Summary of Scientific Contributions

All scientific contributions of this thesis are related to the tasks described in Section 1.1.1 and Section 1.1.2. We propose a framework that encompasses methods to identify, derive and codify relevant knowledge from evolving objects w.r.t. a target concept. Further, auxiliary contributions that help to promote usability of the methods and reliability of the results, such as parameter selection heuristics and cluster-based feature subspace validation, are incorporated.

To summarize, in this thesis we make the following contributions:

FRAME: We propose a framework that takes as input a dataset of objects described by short sequences of historical records and that identifies relevant dimensions.

- **FRAME.DERIVE:** The first part of the framework encompasses methods to derive and codify the implicit dimensionality from the object's evolution given by their historical records:
  - **FRAME.DERIVE.PATTERN:** We present a method to derive relevant patterns and codify them into static features from the sequential values of single time-series features [27], [28].
  - **FRAME.DERIVE.EVO:** We propose a method to codify the evolution of objects and groups of objects in the feature space and in relation to their peers [29].

- **FRAME.IDENTIFY:**The second part of the framework encompasses methods to identify those features, from the set of features used to describe the objects, that are relevant regarding the concept / target variable:
  - **FRAME.IDENTIFY.SELECT:** We present a constraint-based subspace selection algorithm that identifies a relevant feature subspace with the help of expert knowledge or small quantities of class labeled data [30], [31].
  - **FRAME.IDENTIFY.VALIDATE:** We show a technique to validate the relevancy (w.r.t. to the classification problem) of feature subspace regions on independent datasets to promote reliability of our method's results. These validated regions represent the basis for the feature subspace selection. [31].
- **AUX:** We incorporate heuristics to select parameters of underlying base-methods to promote usability of our methods in different domains [29],[30],[31].

Table 1.1 links the aforementioned contributions to the thesis sections, publications and research questions.

Contribution	**Sections**	**Publications**	**RQ**
FRAME	Chapter 2	/	Core RQ
FRAME.DERIVE	Chapter 4	[27],[28],[29]	RQ 1
FRAME.DERIVE.PATTERN	Section 4.3	[27],[28]	RQ 1.1
FRAME.DERIVE.EVO	Section 4.4	[29]	RQ 1.2
FRAME.IDENTIFY	Chapter 5	[30],[31]	RQ 2
FRAME.IDENTIFY.SELECT	Section 5.3	[30],[31]	RQ 2
FRAME.IDENTIFY.VALIDATE	Section 5.3.5	[31]	RQ 2
AUX	Section 4.3.1	[31]	RQ 1/2

Table 1.1: Summary of scientific contributions which are linked to the author's publications, thesis sections and research questions.

1.4 Outline

This thesis is divided into six chapters. This first chapter consists of an introduction and motivation of the thesis topic, a summary of the research questions and an overview about the scientific contributions. Chapter 2 introduces notions that are used throughout the thesis and connects and places our contributions in a broader framework for the exploitation of background knowledge to identify relevant dimensions of evolving objects. In Chapter 3 we discuss the related work w.r.t. to the different parts of our proposed contributions and provide required background literature that is necessary for an understanding of this thesis. In Chapter 4 we show the first part of our framework, EVO-Extract, which is used to exploit different information from the evolution of objects to derive and codify new features that are relevant w.r.t. a target concept. This chapter contains parts of our papers [27–29]. On the other hand, Chapter 4 discusses the second part of the framework, STATIC-Select, that identifies subsets of relevant dimensions that are explicitly given to describe objects with the help of similarity constraints. This chapter contains parts of our papers [27, 30–32]. The last part of our thesis Chapter 6 concludes our work and provides an overview of results, answers to the research questions, limitations and an outlook into potential future work. Reused parts of the thesis are indicated at the beginning of a chapter with a reference to the respective source.

2. A Framework to Identify Relevant Dimensions of Evolving Objects

This chapter provides a general overview and definitions regarding the proposed framework components, objects and functions. The overall framework is a vessel to answer the core research question:

How to find relevant dimensions of evolving objects for classification tasks with the help of background knowledge?

Initially we will define the notions used throughout our proposed framework. Afterwards, we give a general overview about the framework components EVO-Extract, for the extraction of features associated with the evolution of objects, and STATIC-Select, for the selection of relevant feature subsets. Here, each part of the framework addresses one or more of the research subquestions as stated in Section 1.2. At last we conclude this chapter with a definition of a decision basis that can be used to decide if the core research question is positively answered.

Parts of Chapter 2 are from our previous publication [31] (with modifications).

2.1 Formal Specifications

In the first section we define the basic objects of our framework. We first define what an evolving object is and the data scheme used for its representation. Then the notions of subspace clustering, instance-level constraints as well as the distance function we use for our similarity calculations during clustering and during the similarity assessment between (heterogeneous) objects are defined.

2.1.1 Evolving Objects and Data Scheme

In this thesis we are dealing with evolving objects. Evolving objects, which we hereafter label as ”instance”, are described by a set of static features and a set of time-series features. We denote as F the set of static features and G as the set of time-series features associated with a dataset D. Each static feature is only observed once so that for each instance exactly one or no value (missing value) exists. The value of an instance $x \in D$ for the i-th's static feature $f_i \in F$ is given by $f_i(x)$. In contrast, each time-series feature may be observed one or multiple times. Therefore, each instance can have multiple values for a single time-series feature (one per observation). We denote the t-th's observation of the i-th's time-series feature $g_i \in G$ as $g_{i,j}$ and the value of an instance x for the t-th's observation of g_i as $g_{i,t}(x)$. We further denote the set of observations of the i-th's time-series feature as $G_{i,\cdot}$, the set of t-th's time-series feature observations as $G_{\cdot,t}$ and the projections of a set of instances C onto the feature-(observation) space induced by $G_{i,\cdot}$ as $\pi_{G_{i,\cdot}}(C)$ (with $\pi_{G_{i,\cdot}}(x)$ being the projection of an instance x onto $G_{i,\cdot}$). Figure 2.1 shows an abstraction of the feature scheme for both feature sets as well as the observation sets for time-series features.

Finally, each instance is associated with a single class. The class variable is denoted as *class* and an instance x's class-label is denoted as $class(x)$. Note that each instance has exactly one class label, i.e. the *class* is only observed once / is static for each instance.

We further define the notion of dimensionality as follows:

Explicit dimensionality: Let n denote the number of time-series features. The explicit dimensionality of an evolving object x, is given by the complete set of it's static feature-values $\{f(x)|f \in F\}$ together with the set of

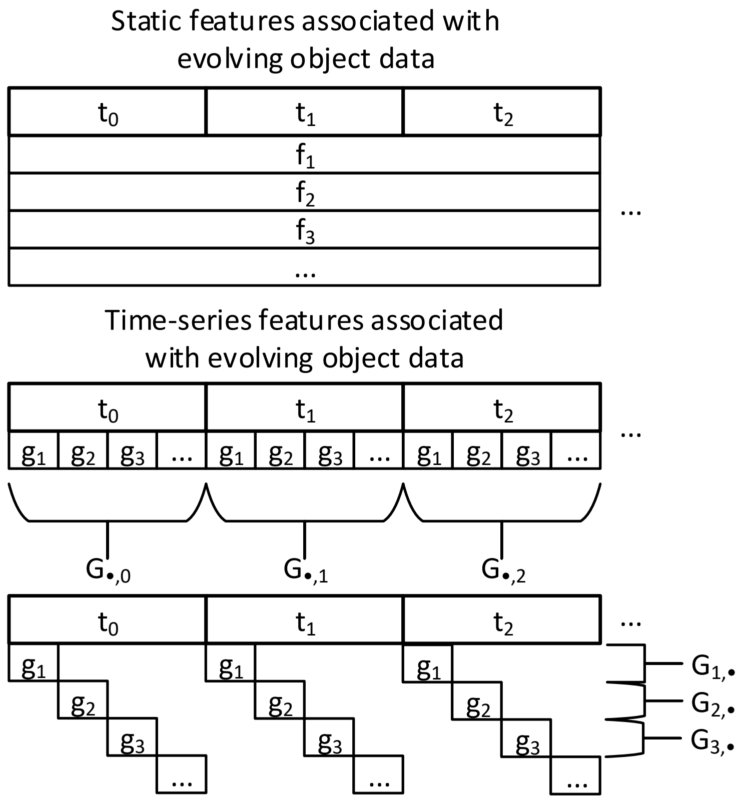

Static features associated with evolving object data
t0
t1
t2
f1
f2
f3
...
Time-series features associated with evolving object data
g1
g2
g3
G•,0
G•,1
G•,2
G1,•
G2,•
G3,•

Figure 2.1: Feature schemes for static and time-series features. For time-points (time-frames/time-windows) $t_0, t_1, t_2, ...$ the associated static features $f_1, f_2, f_3, ...$ never change values for the objects. In contrast, features $g_1, g_2, g_3, ...$ are observed once for each t, implying that values for objects may be different in each time-frame.

all observed time-series features-values $\{g_{i,t}(x)|g_{i,t} \in \cup_{i=1}^{n}(G_{i,\cdot})\}$. Therefore, explicit dimensions are given by the sets F and $\cup_{i=1}^{n}(G_{i,\cdot})$.

Implicit dimensionality: Let n denote the number of time-series features. The implicit dimensionality of an evolving object x, is given by the changes, and derivable statistics w.r.t. those changes, in the observed time-series values $g_{i,0}(x), g_{i,1}(x), ..., g_{i,t}(x), \forall i = 1, ..., n$. Additionally, the implicit dimensionality includes the object's x relation to *similar* objects w.r.t. the aforementioned changes, statistics, and to changes in the explicit dimensionality of those similar objects to x, whereas similar objects may be cluster peers of x or neighbors. Therefore, implicit dimensions are hidden in the evolution of objects over time and the objects relation to each other.

2.1.2 Subspace Clustering with Constraints

One component proposed in this thesis works on techniques used in both subspace clustering and constraint-based clustering in order to find groups of similar instances w.r.t. a target concept and derive relevant feature sets from these clusterings. Because datasets often contain many features and are therefore high-dimensional, this dimensionality may dilute clusters that could be found in subspaces of lower dimensionality. When such clusters contain instances that are similar w.r.t. our target concept, subspaces that lead to the identification of such clusters contain *relevant* (w.r.t. the concept) information. To find such subspaces, subspace clustering methods that are dedicated to the detection of groups of similar data points in subspaces of the complete dataset are required. For this, we define the term subspace as follows:

Subspace: Given a dataset D with associated static feature set F, any subset $F' \subseteq F$ is a subspace.

Based on this, the projection of D onto F' is denoted as $\pi_{F'}(D)$, i.e. the set of instance projections. Then, the goal of subspace clustering is the discovery of clusters $C \subseteq \pi_{F'}(D)$ in at least one F'.

However, utilizing only subspace clustering methods is not feasible if the goal is to find potential *relevant* clusterings. A subspace search algorithm without guidance detects any groups of instances that are similar in any feature subsets. For example, for the target concept ”fatty liver”, the shoe size of patients is completely unrelated to the exhibition of this disorder. However, in this case using plain subspace clustering methodologies could lead to the identification of clusters where patients with similar shoe size (or other unrelated features) are contained. When looking for subspaces that are informative towards a target, such an approach leads to an enormous space of irrelevant results and high complexity.

To guide the subspace search algorithm w.r.t. the target concept we therefore incorporate principles of constraint-based clustering in one of our components. We utilize two types of instance-based constraints: Must-Link

(ML) and Not-Link (NL) constraints which should contain information w.r.t. the similarity of objects. In this thesis, we define a ML and NL constraint as follows:

Constraint: A constraint $CON \in \{ML, NL\}$ in D is a set of two instances $\{x_1, x_2\} \subseteq D$, with $x_1 \neq x_2$.

Both types of constraints differ in their definition of satisfaction:

Constraint satisfaction: Let a clustering of n clusters $\mathcal{C} = \{C_1, ..., C_n\}$ be discovered in dataset D. Let further $satisfaction(CON, \mathcal{C})$ be a function on the satisfaction of a constraint CON in $\mathcal{C}$. A Must-Link constraint $ML = \{x_1, x_2\}$ is called satisfied if both instances under constraint are member of the same cluster in the clustering:

$$satisfaction(ML) = \begin{cases} 1 & \text{if } \exists C \in \mathcal{C} : \{x_1, x_2\} \subseteq C \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Contrary a Not-Link constraint $NL = \{x_1, x_2\}$ is called satisfied if both instances under constraint are not member of the same cluster in the clustering:

$$satisfaction(NL) = \begin{cases} 1 & \text{if } \neg\exists C \in \mathcal{C} : \{x_1, x_2\} \subseteq C \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Given these constraints definitions, the Must-Link $\mathcal{ML}$ and Not-Link $\mathcal{NL}$ sets of constraints are supersets of Must-Link and Not-Link constraints.

Figure 2.2 shows an example for two subspace clusterings with one ML and NL constraint.

2.1.3 Distance Computations

Many of our methods require that distance calculations are executed. Since we cannot assume that the features of objects are scaled uniform, we opt to use the Heterogeneous Euclidean Overlap Metric (HEOM) [33] distance function (across all our methods) that is able to handle objects comprised of differently scaled features, i.e. a distance function that can handle both nominal and continuous features as well as missing values. Let $f(x_1)$, $f(x_2)$ be the values of feature $f \in F$ for participants $x_1, x_2 \in D$. Then:

$$HEOM(x_1, x_2) = \sqrt{\sum_{f \in F} \delta(f(x_1), f(x_2))^2} \tag{2.3}$$

where

$$\delta(f(x_1), f(x_2)) = \begin{cases} 1 & \text{if } f(x_1) \text{ or } f(x_2) \\ & \text{are missing,} \\ olap(f(x_1), f(x_2)) & \text{if f is nominal,} \\ diff(f(x_1), f(x_2)) & \text{if f is continuous.} \end{cases}$$

with

$$olap(f(x_1), f(x_2)) = \begin{cases} 0 & \text{if } f(x_1) = f(x_2), \\ 1 & \text{otherwise.} \end{cases}$$

and

$$diff(f(x_1), f(x_2)) = f(x_1) - f(x_2).$$

Features must be normalized into interval $[0, 1]$ beforehand when utilizing Equation 2.3. In the following chapters, we always use the original HEOM when calculating distances. In the case that our methods apply the HEOM on objects or spaces that are defined differently to the ones defined in this section, we indicate the modifications to the HEOM.

2.2 Framework Components

The framework is depicted in Figure 2.3. It consists of two main components, EVO-Extract and STATIC-Select, as well as two auxiliary components for data preprocessing and cluster relevance validation. Optional components and data flows are depicted with dashed lines. The framework expects a dataset as input according to the scheme described in Section 2.1.1:

Discovery dataset: We denote the expected dataset as the discovery dataset because here, the expert wants to learn what implicit dimensionality exists and which dimensions are relevant. The discovery dataset is a set of

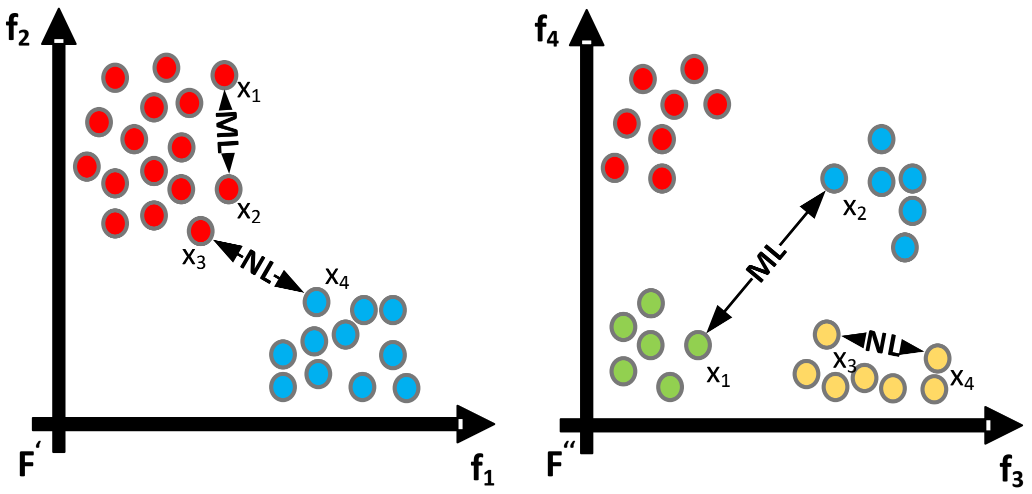


Figure 2.2: Example clusterings of the same objects in two different subspaces $F' = \{f_1, f_2\}$ and $F'' = \{f_3, f_4\}$, with the set of Must-Link constraints $\mathcal{ML} = \{\{x_1, x_2\}\}$ and the set of Not-Link constraints $\mathcal{NL} = \{\{x_3, x_4\}\}$. Object color represents cluster membership. In F' all constraints are satisfied because x_1, x_2 are in the same and x_3, x_4 are in different clusters. In F'' both constraints are unsatisfied because the aforementioned conditions do not hold true.

evolving objects were relevant features are constructed and identified (in comparison to the validation dataset that is optional and can be used to validate findings found in the discovery dataset).

The discovery dataset first undergoes a **preprocessing step** where adjustments to feature-value ranges are conducted. Preprocessing is required so that our methods can be applied. The preprocessing is described in Section 4.1.

Then, the preprocessed discovery dataset is input to the **EVO-Extract component** (cf. Chapter 4). EVO-Extract applies our methods on the discovery dataset to derive three types of static features, so-called ”Evolution Features”, which contain information from the observed evolution of objects within the dataset. Here, EVO-Extract conducts three activities / executes three methods to derive these features:

Deriving pattern-based Evolution Features: These features are based on relevant patterns observed in the evolution of objects (ground-truths and group-based). The respective method is proposed in Section 4.3.

Deriving object-based Evolution Features: These features are based on the intermediate individual evolution of an object (both unsupervised and ground-truths object-based without considering peers). The respective method is proposed in Section 4.4.

Deriving group-based Evolution Features: These features are based on the intermediate evolution of complete object groups (both unsupervised and ground-truths based). The respective method is also proposed in Section 4.4.

The output of EVO-Extract is the discovery dataset enriched with the derived Evolution Features.

The second main component, **STATIC-Select** (cf. Chapter 5), expects any static discovery dataset (i.e. a dataset that only contains a static set of features) with either labeled training data or, in the case that no training data exists, separate sets of associated instance-level constraints. If the EVO-Extract output or any other evolving dataset is used, it must be first transformed to a static dataset with the preprocessing component. STATIC-Select can then be used to identify relevant feature sets, using either of the following two methods:

Traditional feature selection (CFS-based): If a substantial amount of labeled training data exist, a traditional feature selection method is applied. The method is based on the Correlation-based Feature Selection (CFS [12]) and is described in Section 5.2.

Constraint-based feature subspace selection: If labeled data does not exist our constraint-based subspace selection technique can be applied. However, our method may also be used when labeled training data is scarce or only instance-level constraints are given. Our method is proposed in Section 5.3.

Output of STATIC-Select is a set of relevant features. Because our constraint-based subspace selection technique evaluates feature subspace relevance according to the relevance of identified subspace clusters, our framework encompasses an optional validation component:

Validation component: The validation component helps users to assess the transferability / generalization of relevant feature subspace clusters w.r.t. a second dataset. This helps to promote confidence in the findings. Our validation component is proposed in Section 5.3.5.

To use this component an additional **validation dataset** is required and our constraint-based subspace selection technique must be utilized in STATIC-Select. If these conditions are satisfied, the validation component matches the discovery and validation dataset (based on features specified by the user) and validates whether relevant regions in feature subspaces that were found in the discovery dataset are also relevant in the validation dataset (cf. Section 5.3.5).

Note that both main components, EVO-Extract and STATIC-Select, can be used independently and do not require each other. If a static discovery dataset is given and the goal is to find relevant features in it, then STATIC-Select may be used independently. In contrast, if one wants to apply their own method for the selection of relevant dimensions, then only EVO-Extract may be applied. In the chapters Chapter 4 and Chapter 5, we show the two main components EVO-Extract and STATIC-Select. In the EVO-Extract chapter a brief description of the preprocessing component is included. The STATIC-Select chapter includes the cluster relevance validation component.

2.3 Core Research Question Refinement and Formalism

Our complete proposed framework serves to answer our core research question:

How to find relevant dimensions of evolving objects for classification tasks with the help of background knowledge?

The two existing cases associated with *"finding relevant dimensions for evolving objects"* that must be investigated to derive an answer to this question are the following:

Case 1: Dimensions are not yet codified as features

Given the evolving objects, identify relevant dimensions implicit to the object evolution that lead to a performance increase in the respective classi-

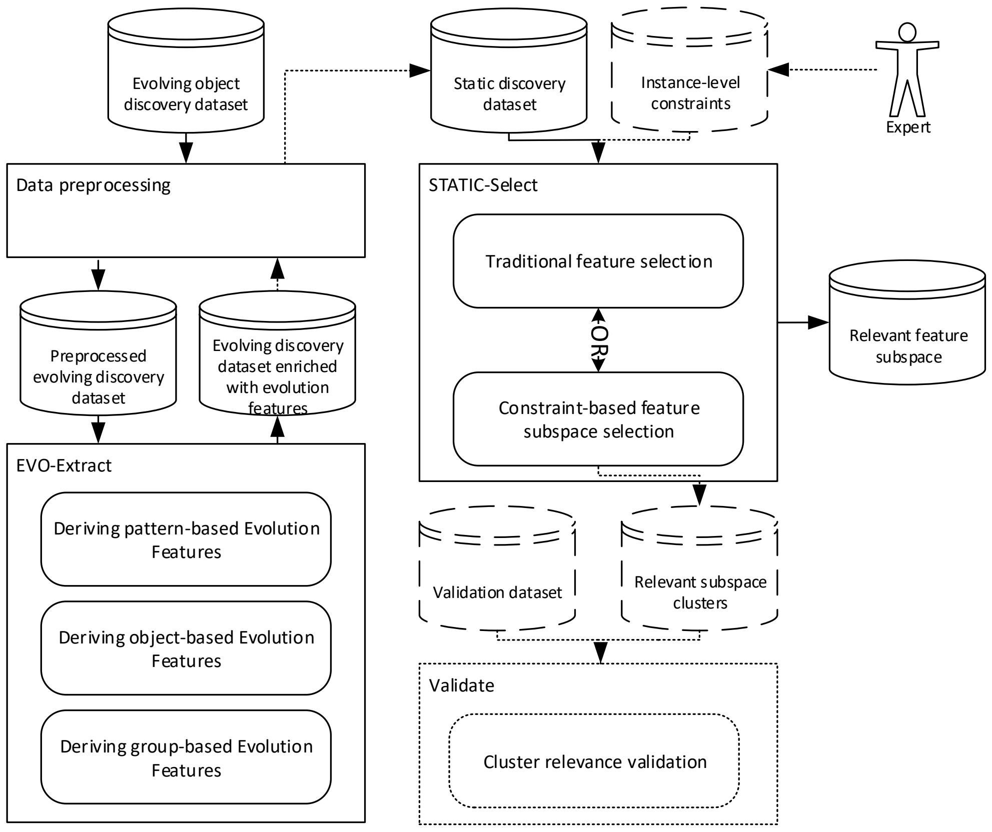

Evolving object discovery dataset
Static discovery dataset
Instance-level constraints
Expert
Data preprocessing
STATIC-Select
Traditional feature selection
OR
Constraint-based feature subspace selection
Relevant feature subspace
Preprocessed evolving discovery dataset
Evolving discovery dataset enriched with evolution features
EVO-Extract
Deriving pattern-based Evolution Features
Deriving object-based Evolution Features
Deriving group-based Evolution Features
Validation dataset
Relevant subspace clusters
Validate
Cluster relevance validation

Figure 2.3: Overview of our framework to derive and identify relevant dimensions of evolving objects. Optional components in dashed lines.

fication task when added to the original set of explicitly available dimensions. In the first case, the relevant dimensions are not explicitly available and must be first derived and then codified. This case is summarized by RQ 1 (cf. Section 1.2) and the first framework-component, EVO-Extract, should provide solutions to it. Related literature w.r.t. case one is found in Section 3.4.2 (regarding existing methods) and Section 3.1 (regarding underlying methods we use to tackle case one).

Case 2: Dimensions are already codified as features

Given a set of dimensions, decide on a subset of dimensions that are relevant, i.e. a subset of dimensions that performs equally good or better than the original set of dimensions in the classification task. In the second case, the dimensions are already given. This case is summarized by RQ 2 (cf. Section 1.2) and the second framework-component, STATIC-Select, should provide solutions to it. Related literature w.r.t. case two is found in Section 3.3 (regarding existing methods) and Section 3.5 (regarding underlying methods we use to tackle case two).

Considering these two cases and the necessity for the exploitation of different types of background knowledge (as specified in the core research question), we define the following formalism to decide on a positive answer to the core research question:

Let $expr(RQ1)$ and $expr(RQ2)$ define logical expressions that are $True$ if RQ 1/RQ 2 are answered positively.

$$expr(RQ1) \vee expr(RQ2) \tag{2.4}$$

Expression 2.4 is true if at least either RQ 1 or RQ 2 are answered positively. Then, we consider the core research question as *"positively answered"* if, and only if, the logical expression 3.4 holds true, and if the associated component(s) that led to this result utilized at least two different types of background knowledge to achieve this (i.e. if both RQ 1 and RQ 2 are answered positively, the respective framework components catering to these questions successfully utilized at least two different types of background knowledge). The expressions $expr(RQ1)$ and $expr(RQ2)$ will be defined in the respective component chapters.

3. Background & Related Work

In this chapter we provide an overview about work related to this thesis. We provide background knowledge which is necessary to understand our scientific contributions and position our research in the context of existing work. Apart from the identification of relevant dimensions, background knowledge is utilized in a plethora of different data mining tasks and mainly for non-evolving objects. The most common kind of background knowledge, information on the true class label of objects, is necessary for classification algorithms to induce classification models like k nearest neighbors (knn) or decision trees and is input for feature selection methods. Constraints on the other hand, are mainly used for controlling the clustering process and generate a desired grouping, but they can also help in the identification of high-quality feature subspaces. Background knowledge about historical recordings is used to generate meta features and relevant patterns for classification. Because many of our methods utilize clustering techniques, we also discuss related work on basic unsupervised clustering methods in the next section and the related literature regarding the different kinds of background knowledge (including constraint-based clustering) in the sections thereafter.

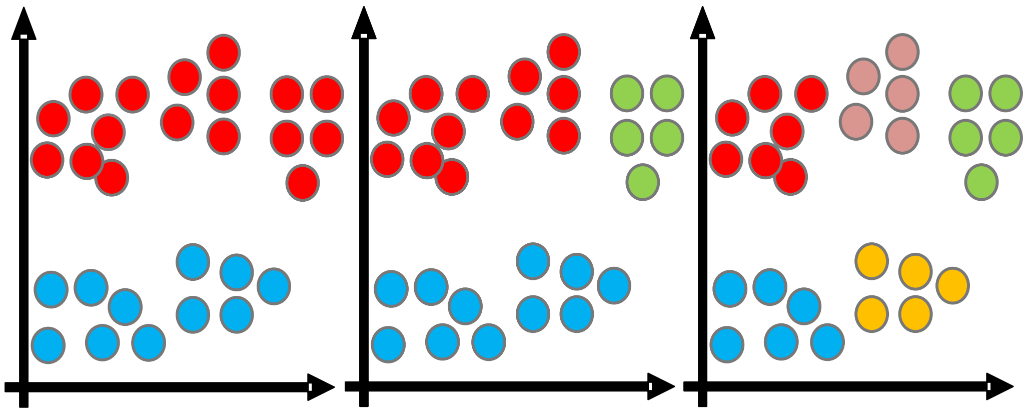


Figure 3.1: Different clusterings for the same set of objects in an arbitrary two dimensional space. Colors refer to clusters.

3.1 Clustering

The methods we propose in this thesis utilize techniques and algorithms from different areas of clustering. Clustering is recognized as one of the main data mining techniques and aims at grouping similar objects into the same cluster and dissimilar objects into different clusters [34]. Clustering is widely utilized to group objects in many different domains such as patients in health care data [35], e-mails for spam detection [36] or credit card transactions to identify fraudulent ones [37]. The concept of a cluster is not uniform [38]. Given a set of objects, different clustering algorithms may identify different clusters with varying shapes and constitution. Figure 3.1 shows alternative clusterings for the same set of objects.

One reason for this is that basic clustering algorithms are completely unsupervised and use only the internal structure of an object for clustering without exploiting any background knowledge regarding a (potential) true cluster membership of objects [34]. Typically, the objects themselves are described by n-dimensional feature vectors and to assess the required similarity between objects/representatives, domain-specific distance/similarity functions are utilized. Many taxonomies for clustering techniques have been proposed [39], [40], [41]. Figure 3.2 shows an exemplary classification of clustering families into hierarchical, partitioning relocation, density-based and grid-based methods [40]: Traditional hierarchical methods construct a hierarchy of clusterings. They work iteratively and are either agglomerative (bottom-up) or divisive (top-down) [42]. Bottom-up methods start by declaring each object as a one-object cluster. They merge in each iteration the two most "‘appropriate"’ clusters. In contrast, divisive methods define a single cluster that encompasses all objects and iteratively split the most "‘appropriate"’ cluster. Appropriateness may be defined by similarity metrics, cluster validation metrics etc. Methods terminate when a stopping criterion is reached (for example the number of clusters).

Partitioning relocation methods construct a clustering by iteratively relocating points between several clusters according to an optimization function. In contrast to the many different clusterings hierarchical methods build, partitioning relocation methods produce only a single clustering where the complete data is divided into a set of clusters. Famous examples for this family of algorithms are K-Means [43] and K-Medoids [44], with their more recent extensions Cartesian K-Means [45] and K-Medoids based on step increasing and optimizing medoids [46].

Density-based methods do not assume a specific cluster shape but rather identify dense regions within the feature space. Apart from some notable exceptions (i.e. DENCLUE [47]), methods utilize concepts like *local density* and *connectivity* where each dense region is considered as a cluster. Dense regions are defined by discovering objects that exhibit a "‘high"’ number of other objects in their neighborhood and iteratively checking whether the neighbors are dense too. Utilizing such an approach leads to the discovery of clusters that "grow" in the direction the density of objects dictates. Famous examples for such algorithms include DBSCAN [48] and OPTICS [49]. Despite their age, algorithms like DBSCAN are still widely utilized. Recent advances concentrate on enhancing these algorithms for an application in different domains and scenarios (for example AnyDBC [50] for large complex datasets).

Grid-based methods partition the feature space into segments described by value-ranges of the constituting features. In essence, these methods discretize the continuous features and afterwards analyze the points that fall into these cells (multidimensional bins). Grid-based algorithms often exhibit similarities with methods of the other categories. For example BANG [51] uses a grid-based data view and hierarchical clustering to identify clusters, whereas GCHL [52] combines a density-grid based clustering in combination with an axis-parallel binning scheme.

Figure 3.2 shows the base clustering categories. Some of our proposed methods utilize the density-based clustering algorithm DBSCAN [48] as a base clusterer which falls into the density-based connectivity clustering category. In the following we use the notions defined by Ester et al to describe the DBSCAN algorithm [48]: DBSCAN identifies dense regions in the data space and marks them as clusters. It uses the parameters *eps* and *minPts*. The *eps* parameter defines the neighborhood around an object, and *minPts* defines the minimum number of neighbors a data point must exhibit to be considered a core point. This means that given a point x, dataset D and distance function $dist()$, the *eps*-neighborhood is defined as $N_{eps}(x) = \{y \in D | dist(x, y) \leq eps\}$. If and only if $N_{eps}(x) \geq minPts$, then x is considered a core point. In the case that x exhibits fewer points, it is considered a border point if x is in the *eps*-neighborhood of a core point, i.e. $\exists y \in D | x \in N_{eps}(y) \wedge N_{eps}(y) \geq minPts$ holds true. Otherwise x is a noise point and belongs to no cluster or an arbitrary outlier cluster. With this definition in mind, clusters are build by exploiting the notion of connectivity between core point. The neighborhoods of each set of inter-connected core points define a dense region and form a cluster. This inter-connectivity is defined with the notions of *direct density-reachability* and *density-reachability*. A point y is directly density-reachable from x if and only if $y \in N_{eps}(x)$ hold true and x is a core point. Based on this, a point y is density-reachable from x if there exists a chain of data points $x_0, x_1, ..., x_n$ where each x_i is directly density-reachable from x_{i-1}, with $i \in \{1, ..., n\}$ and it holds true that $x_0 = x$ and $x_n = y$. Now, each dense region / cluster can be identified by selecting any core point and traversing the region via the density-reachability definition. Given a core point, all points that are density-reachable from it belong to the same cluster (including the core point itself). Note that border points may be density-reachable from core points of different clusters. In this case, simple heuristics like random cluster assignment or first-visit cluster assignment can be used.

DBSCAN exhibits a number of advantages over other clustering algorithms: outliers which might have special characteristics can be detected. Additionally, the number and size of clusters is not required to be specified as a parameter. Further, DBSCAN is not limited to linear boundaries but dis-

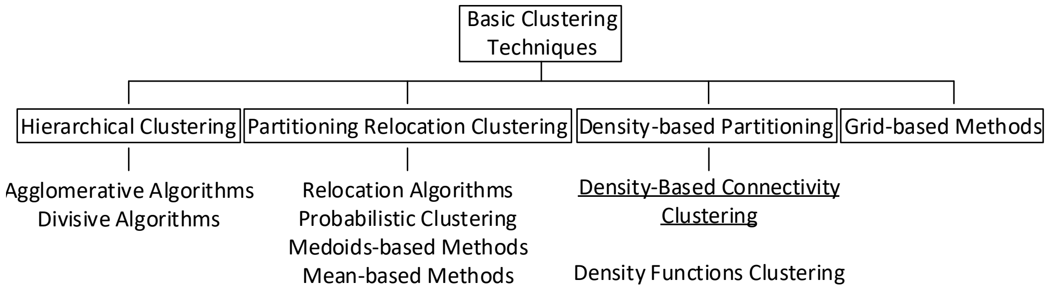


Figure 3.2: Example taxonomy of basic clustering techniques related to the categorization in [40], omitting more advanced and application specific techniques like constraint-based clustering or subspace clustering (thus disqualifying them as ”basic”). Note that no uniform taxonomy exists.

covers arbitrarily shaped clusters, so that no assumptions on the feature's distributions within a dataset have to be made. Figure 3.3 shows arbitrary shaped cluster examples found via DBSCAN.

3.2 Subspace Clustering

One of our contributions within this thesis is based on principles of subspace clustering. Unlike regular clustering methods, subspace clustering techniques aim to find clusters within datasets on subsets of the original dimensions (see e.g. [53]). More formally, let us call any $F' \subseteq F$ a subspace, where D is the dataset (the original dataset) and F is the associated set of features. Let us further denote as $\pi_{F'}(x)$ the projection onto F' for an object $x \in D$ and the set of all object projections as $D_{F'}$. Then, the goal in subspace clustering is the discovery of clusters $C \subseteq D_{F'}$ in one or more subspaces F'.

A key issue of regular clustering methods is that given a high-dimensional dataset, clusters may only exist in subspaces of the data. Such clusters may go completely unnoticed if all the available dimensions are taken into account by diluting the information regarding which sets of characteristics the objects are actually similar in (features that describe the clusters) with irrelevant noise (features that are irrelevant to a cluster).

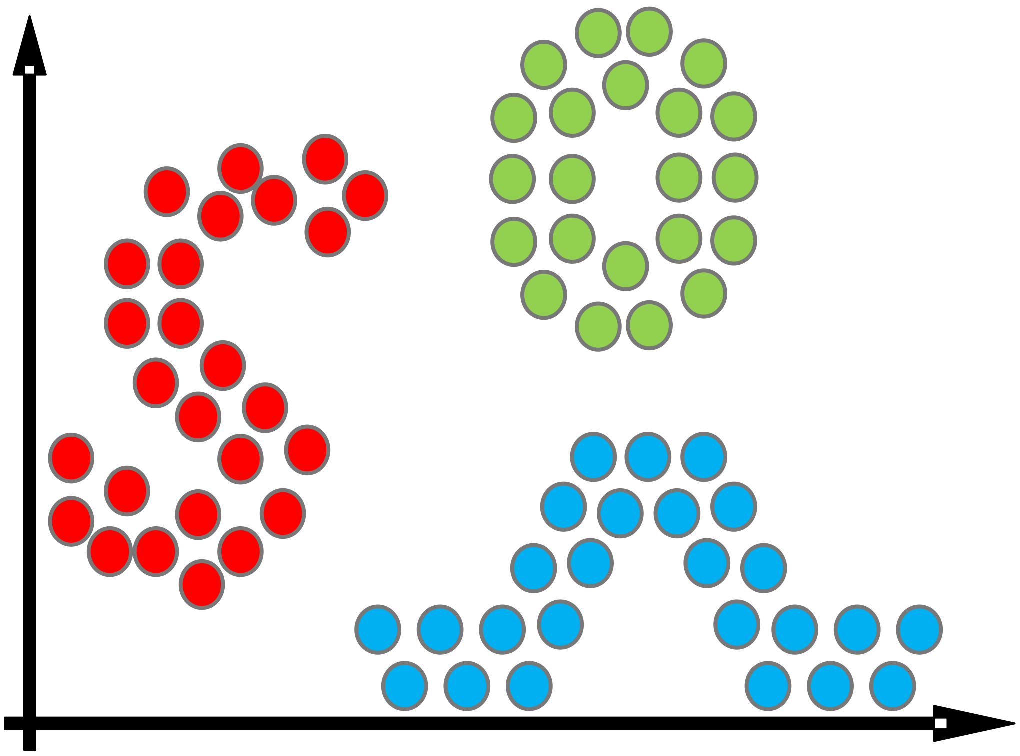


Figure 3.3: Examples of different shapes found by a DBSCAN clustering in an arbitrary two-dimensional space. Colors refer to cluster membership.

Subspace clustering algorithms basically consist of a clustering component that conducts the grouping of the objects and a subspace search component that determines the subspace(s) for the clusterings. They can be differentiated according to their utilized subspace handling strategy into bottom-up and top-down algorithms [54]: Bottom-up algorithms start to identify subspaces of cardinality one that contain clusters and iteratively try to identify clusters in higher cardinality subspaces. Often, they expand the cardinality one subspace clusters with other subspaces that contain clusters and check whether the clusters still exist. To efficiently prune the search space and reduce the combinatorial complexity associated with this process, algorithms of this family employ a monotonicity criteria. For example, in a density-based clustering setting they may assume that if a cluster in a subspace F' of cardinality *cdl* is not dense in a superspace $F'' \supset F$ with $|F''| = cdl + 1$ anymore, than the cluster will not be dense in any superspace of F'', and therefore this branch of the search space can be discarded (i.e.[55]). Members of the bottom-up algorithm category are CLIQUE [56] and MAFIA [57]. Top-down algorithms on the other hand start by either (iteratively) clustering on the full set of dimensions and afterwards determine which dimensions are actually relevant to each cluster (cf. [53]) or by identifying a subset of dimensions for a given set of instances, so that a clustering criterion is satisfied (cf. [54]). Both approaches come with some significant drawbacks. Top-down methods can exhibit high complexity when multiple clustering on the complete feature space are required. Further, to reliable detect cluster members in high-dimensional spaces goes against the assumption that clusters can be completely diluted by irrelevant dimensions. To overcome such problems, algorithms may utilize diverse sampling strategies [54],[53]. Two example top-down subspace clustering algorithms are PROCLUS [58] and FINDIT [59].

3.3 Exploiting Labeled Training Data

Ground truth (as given by class-labeled training data) is used in different data mining tasks. Let D be a dataset and c a class variable. Each $x \in D$ describes an evolving object with x being a multi-dimensional feature-vector and each object is assigned one class-label $c(x)$. Also, let $D_{tr} = \{x' \in D | c(x') isknown\}$ be the subset of objects from D with known class. If in this case the class-variable describes the target concept that is associated with the data mining task, then D_{tr} can be utilized as the labeled training data or ground truth (cf. [60]). Areas that are related to the work presented in this thesis, where training data is used as background knowledge, are classification (for static and evolving objects) and feature selection. We will discuss related work on those topics in the following subsections.

3.3.1 Supervised Feature Selection

Using labeled training data to find relevant dimensions is a well studied topic in the research field of feature selection. Datasets used in classification are not necessarily suited for a classification task out-of-the-box. They may contain a plethora of features with redundant or limited information regarding the concept that the classifier aims to model. In "Feature Selection", associations between the training objects feature-values and the class variable are evaluated to identify sets of relevant (w.r.t. a target concept) and non-redundant features [12]. The goals of these techniques are to improve the predictions build on a dataset, possibly improving their speed and cost-effectiveness, and enhancing the understanding of the underlying data [61]. Existing algorithms fall into one of three categories: Wrapper-methods directly derive the quality of a feature set from the estimated performance of classifiers [62]. Filter-methods use a separate criterion to evaluate feature sets, for example the average correlation with the target variable used in Correlation-based Feature Selection (CFS) [12]. Finally, embedded methods integrate the feature selection and classification steps [61]. Recent methods concentrate on distinct application fields like feature selection in medical domains [63] or multi-view feature selection [64]. Many algorithms lack out-of-the-box applicability on the kind of real-world data we deal with. Hence, in our setting an extension of traditional methods that can deal with differently scaled features, unavailability of class label data and multiple observations for each feature is required to find the relevant dimensions.

3.3.2 Semi-Supervised Feature Selection

Traditional feature selection algorithms use the labels of the training data to decide on the relevance of features. They typically do not consider the unlabeled data and may perform poorly when labeled data is sparse. Differently, semi-supervised feature selection algorithms try to "best" exploit both the given labeled and unlabeled data. The traditional assumption for semi-supervised feature selection is that the labeled sample is an unbiased sample of the overall population [65] (which may not always hold true [66]). Available semi-supervised approaches have no uniform taxonomy. They differ widely in the actual handling of both the labeled and unlabeled data. One possibility for a categorization of these methods is to differentiate them w.r.t. the type(s) of supervised knowledge they use:

Class-label based semi-supervised methods compute the relevancy of feature sets w.r.t. the class variable on the labeled data and either bootstrap a class [66] on the unlabeled data or calculate redundancy scores with their help. For example, Sechidis et al [66] propose two simple surrogate strategies to test the dependency between a feature f and a binary feature / class-variable *class* (with outcomes ”positive” and ”negative”) and rank features w.r.t. their mutual information with *class* when only limited amount of labeled, and large amounts of unlabeled training data is available. For both strategies, statistical testing (using a G-test on independence) and ranking (on mutual information), they assume that all unlabeled data are either considered as being completely positive or completely negative. They provide a decision-threshold that is based on the amount of ground-truth and a prior belief of the domain-expert on the positive-class probability. The threshold is then used to decide on the class (either all positive or negative) of the unlabeled data for testing and ranking purposes (i.e. decision on which surrogate to use). Authors show that their approach yields competitive results compared to more complex methods. For large data, they report that the surrogate approach yields exactly the same ranking as if all of the data were labeled and used. Additionally, hypothesis-testing with the surrogates results in the same False-Positve-Rate (FPR) but higher False-Negative-Rate (FNR) compared to hypothesis-testing when ground-truth is available for the complete data. Here, the FNR can be decreased by using the provided threshold for the surrogate decision. In comparison, He et al. [67] use a mutual information based approach where the relevancy of attribute sets is calculated on only the labeled data and the redundancy of attributes on both the labeled and unlabeled data.

An alternative approach is the use of instance-level constraint-based semi-supervised methods to decide on a set of most relevant features towards a target concept / class-variable. Such methods use decisions from domain experts (or derive these decisions from other types of background knowledge) on the similarity of objects or variables, instead of class-label data, to guide feature selection algorithms. There exist a number of publications which use the Laplacian Score in conjunction with a Constraint Score (based on constraints on the similarity/dissimilarity of instances) to decide on relevant features [68–70]. Here, the Laplacian Score promotes features with high variance while preserving locality, i.e. features where instances in the same neighborhood have close values in comparison to instances of different neighborhoods. The Constraint Score on the other hand, promotes features where instances under a similarity constraint are close and instances under a dissimilarity constraint observe high distance to each other.

Our in Chapter 5 proposed approach DRESS shares some key ideas with traditional instance-level constraint-based semi-supervised feature selection methods. However, in comparison to DRESS, traditional methods have some key disadvantages. First, they are based on parameters to define neighborhoods that have shown to have a high impact on the method's performance. Second, features are scored and ranked individually. The structure of feature subspaces of a cardinality greater than one is never investigated, thus omitting possible patterns that emerge from the interaction of multiple features. Third, the proposed approaches all exhibit at least three distinct parameters without propositions on how to impute reasonable settings. Keeping in mind that these methods try to deliver relevant feature subspaces when only limited/small quantities of background knowledge is available, an extensive tuning step that requires a substantial amount of training data should be prevented and goes against the application scenario of these methods.

3.4 Exploiting an Object's Evolution

In many data mining tasks, objects are not static and only observed once, but do rather change over time (evolve) and are also observed multiple times during their evolution. One part of our proposed framework deals with the extraction of target-concept relevant knowledge from such evolving objects which are associated with sets of time-series features (univariate time-series) and static features (cf. Section 2.1 for our formal specification of evolving objects). In this section, we present work related to the exploitation of evolving objects in the context of this thesis, which includes work on the abstraction of time-series and the use of such abstractions for classification.

Note that in the case of classification of evolving objects (that may be described by time-series features) we only consider that exactly one label per time-series /evolving object is predicted, regardless of the fact that a label may change over time and could be predicted in different moments.

3.4.1 Time-Series Abstraction

Often, knowledge discovery is concerned with the identification of patterns within datasets and their representation towards the human expert. Pattern representations of data mining algorithms should be close to those used by a human to support the knowledge generation process. For example, the mere information that customers with similar income level buy a specific product more often could be carefully abstracted to tell the expert that this is actually a group of customers with relatively high income. Such abstracted pattern is more natural to use by humans for decision making instead of the plain income numbers. For static data, discretization, clustering, rule-abstraction etc. are used to accomplish this task. In the case of time-series or temporal data, these techniques are not sufficient. While it is possible to cluster or discretize time-series, the additional temporal component can still lead to issues in the representation and understandability of the resultant abstractions. After all, a cluster in static data is much easier to understand, for example it may refer to high-income, low-age customers. In contrast, a cluster in time-series data may also include implicit dimensionality given by the object's trajectory directions, mean or variance. A key task in abstracting time-series is therefore representation. According to Höppner, abstraction methods can be separated into inductive and deductive approaches[20]:

- Inductive approaches produce features (from the evolution of objects) for which the semantic is not known a-priori. For example, clustering algorithms can separate a time-series in similar parts. Afterwards, the clusters can be used for the representation of the time-series. However, the meaning of the feature-values (the clusters) is not known beforehand.
- Deductive approaches use pre-defined features and derive the values for those features from the observed time-series. The abstracted time-series is then represented by a feature-value vector. For example, representing a time-series by its mean, variance and length falls under the deductive category.

An advantage of the inductive approach is its data-driven nature. Assumptions on *how* a time-series should be abstracted are only made implicitly by the choice of the (clustering/segmentation) algorithm, similarity measure and the provided parameter settings. In contrast, the deductive approach provides an abstraction that is more easily interpretable because of the pre-defined features that are used. However, it may miss valuable information that is not covered by those features.

The problem of temporal data or time-series abstraction is addressed by different authors [71–73]. Proposed methods derive simple statistics such as mean or variance from temporal features and use these new representations for predictive tasks. Such representations are easy to interpret but a lot of information may get lost, predictive potential unrealized and relevant knowledge regarding the target concept unrepresented. In contrast, more powerful methods do not take a target concept into account [74, 75], which leads to uninformative representations, or the representations themselves that are used to discriminate between classes are hard to interpret [14, 76].

In this thesis we provide means to derive features from the evolution of instances in a data-driven and pre-defined way. Both approaches are novel: in our data-driven approach we are able to identify relevant, arbitrary shaped patterns of both continuous and nominal time-series features without the requirement of a prior discretization or abstraction. The approach uses clustering with semi-supervision to find an agreement between internal structure of the data (clusters / patterns) and target-concept relevancy (purity of patterns). Our second approach uses pre-defined features that not just encode the evolution of single instances but also the information of their evolution w.r.t. other clusters in the dataset.

3.4.2 Using Time-Series Abstractions for Classification

Whether an inductive or deductive approach is used to abstract time-series, the new representations of the time-series can be utilized in classification workflows. Such an approach has two advantages. First, the resultant representations are often easier to interpret (which is a key asset when besides the classification performance the understandability about how the results were actually achieved is required). Second, certain abstractions lead to static representations (a single feature vector) of the time-series. This enables the expert to use traditional data mining methods that are normally not applicable on time-series data.

Some approaches derive features from time-series (like mean, variance, existence of rules, interval patterns etc.) and build traditional classification models on them [77], [71], [78]. These models can be generated in different ways. (1) Features may either be derived over the complete time-series or

(2) series are first split in time-windows and then features are derived for each window. After extraction, the features of (1) and (2) can be used to build a single classification model (for example a nearest neighbor model). (3) Alternative approaches split the time-series into time-windows and then learn a separate classification model on windows of the series. Models are later used in conjunction as an ensemble to classify time-series instances [72]. Figure 3.4 illustrates these three approaches.

A unique variation to these approaches for classification model generation based on time-series abstraction is the use of shapelets. Shapelets are unique subsequences of the time-series that distinguish the series (and series of the same class) from others. Uniqueness is assessed by calculating the distance between shapelets. A high-quality shapelet should be contained in all instances of the same class (i.e. there exist other shapelets with minimal distance to it in instances of the same class) and not contained in instances of different class (i.e. there are no other shapelets with minimal distance to it in instances of a different class). Extracted shapelets can then be used in Decision Tree based algorithms where a node represents a shapelet that partitions instances based on the observation if they contain a similar shapelet or not [79]. For example, shapelet $shape_a$ may represent a node and is representative of class c. If an instance (time-series) x that should be classified contains any shapelet $shape_b$ that is similar to $shapelet_a$ based on their distance $dist(shape_a, shape_b)()$ and a threshold $thresh$, i.e. $dist(shape_a, shape_b)() < threshold$, x is more likely to be assigned to c. More recent approaches use multiple shapelet trees in Random Forests where instances and shapelet features are randomly sampled to create a more robust tree ensemble classifier and detect the most important shapelets [76].

In this thesis we do not extract shapelets from evolving objects, since we have mixed (continuous and nominal) datasets and consider only short sequences. The features that we derive in this thesis are used in traditional classification workflows. This has the benefit that widely available and thoroughly tested algorithms can be utilized.

3.5 Exploiting Instance-Level Constraints

There are a number of data mining applications where instance-level constraints are utilized. Instance-level constraints mostly come in the shape of

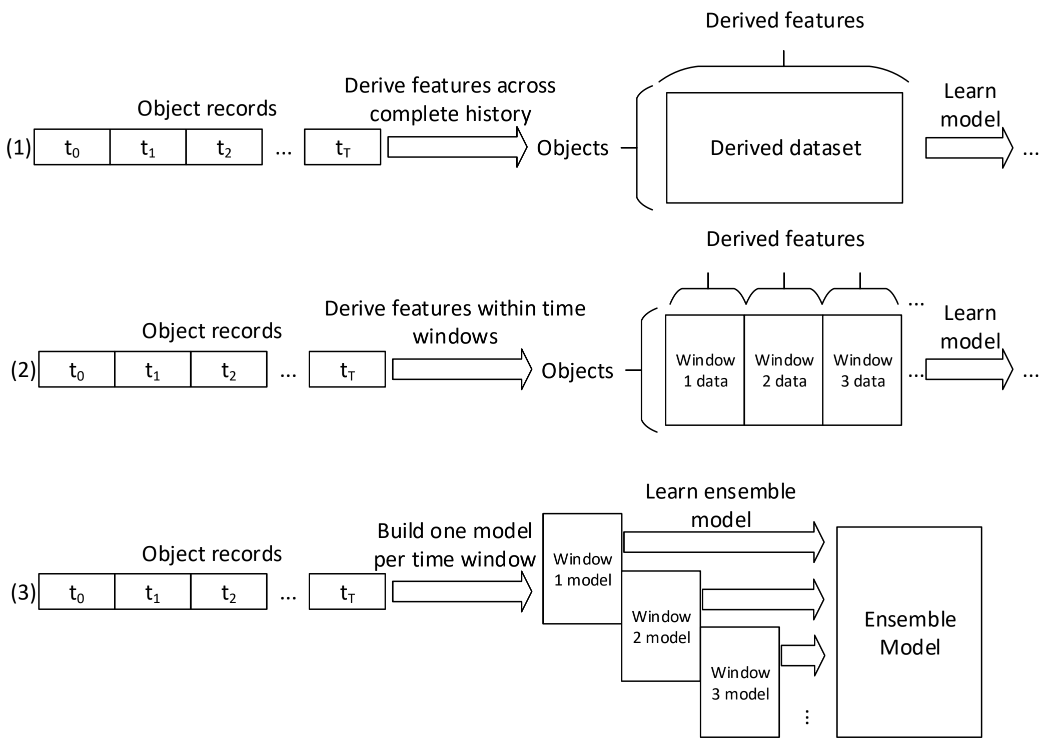


Figure 3.4: Three alternative approaches to learn models for classification tasks from object time-series. In (1) features are derived across the complete object time-series, leading to a single new static dataset that can be used for traditional model learning. Approach (2) first defines time windows over the time-series data and then derives features within each window. This leads to a number of static datasets (one for each window) which are unified into a single dataset that is used for model learning. The third (3) approach is based on (2), but instead of unifying the static datasets of all time windows, one model is learnt on each window. At the end, an ensemble model is generated to classify time-series objects of unknown class.

”Must-Link” and ”Not-Link” constraints between two instances of a dataset. The former type reflects that two instances are somehow similar whereas the latter type reflects that they are dissimilar. What this similarity constitutes is actually up to the domain, algorithm or the expert that defines these constraints. The main application of instance-level constraints-based algorithms is clustering [80]. However, contributions in other areas like classification [81] and distance metric learning exist [82]. Since we utilize constraints in a clustering-based setting, we concentrate on work done on exploiting instance-level constraints for clustering in the following subsections.

3.5.1 Constraint-based Clustering

In constraint-based clustering the aim is again to find groups of similar objects that are also dissimilar to the objects of different groups. However, these methods aim to reduce a drawback of traditional clustering methods: the problem that available knowledge about the true cluster-membership or background knowledge on cluster parameters is not used to guide the clusterer. For traditional methods, the objective functions that are optimized do only take the internal structure of the data into account (i.e. sum of squared errors in K-Means [43]). In contrast, constraint-based clustering may use different kinds of constraints in a dataset D to find clusters $C \in D$ that satisfy these constraints [83]. Constraint-based clustering therefore has the goal to find a clustering that yields groups which agree w.r.t. the internal structure of the data and the external information that is given as constraints, which includes the following constraint-types [21]:

- **Obstacle object constraints** where objects may be modeled as being more or less distant to other objects. For example, a building is only reachable through a bridge which increases the walking distance to this building from a building of the opposite side of the river (i.e. thus increasing the ”shortest distance” to it).
- **Constraints on single objects** where only a subset of all available objects are eligible for clustering.
- **Clustering parameters as constraints** where only specific parameters may be used for a clustering algorithm.
- **Constraints imposed on clusters** where pairs of objects are constrained to be in the same / different clusters.

Especially the last type of constraints, the object-level constraints between pairs of objects, are studied well in literature. For example, Constraint-driven DBSCAN (C-DBSCAN) uses constraints between objects to determine whether to split a cluster or merge two clusters if the constraints disagree with the initial clustering [83]. It utilizes Must-Link (ML) and Cannot-Link (NL) constraints between two objects which either indicate that both objects under constraint should be in the same cluster or different clusters. Another possible approach to use constraints for clustering is to evaluate and choose among different models according to their constraint satisfaction. Dau et al. [84] use a cluster-algorithm independent approach to utilize ML and NL constraints in time-series clustering. Their goal is to determine parameter values for the underlying clustering algorithm or the best clustering among multiple models generated with different parameter values. They first perform multiple clusterings of the data by varying the clustering parameter. After that, the set of all possible object-pairs is created. This set is filtered by removing pairs that would be associated with uninformative constraints (constraints that describe objects with stable conditions), i.e. objects where constraints would be always, or never satisfied regardless of the parameter setting, and object pairs where the constraint satisfaction would be highly oscillating. From the remaining pairs, authors calculate a simplicity value (derived from the number of changes of the constraint satisfaction when putting the objects under a constraint while increasing or decreasing the parameter value). Then, they show the "simplest" pairs to human experts for constraint annotation. Given this annotation, they can then derive the parameter-value(s) and clustering model that satisfies the obtained constraints.

Because of the benefit to guide the clusterer towards a "correct" grouping or to select higher-quality cluster models, constraint-based clustering is utilized in different domains like medical study data to remove confounding factors [85] or in automatic color image segmentation [86].

3.5.2 Constraint-based Subspace Clustering

Fromont et al. are the first authors to consider the integration of constraints, particularly instance-based constraints, into subspace clustering [87]. Constraint-based subspace clustering utilizes constraints to not only find the best clustering in a given dataset according to the preferences set by the user (through the constraints), but also the most appropriate subspace w.r.t. these preferences. For example, consider high-dimensional data about participants of an epidemiological study where for some of them it is known whether they exhibit a specific disease or not. A medical expert now may be interested in groups of similar healthy and ill people, and the characteristics of such groups. To find these groups, the expert could set similarity constraints, according to their desired output and the target-concept, between pairs of healthy, ill and mixed participants. If such an expert would use plain constraint-based clustering it is likely that the high-dimensional data would dilute clusters that are only detectable in lower cardinality subspaces and therefore the expert would not identify relevant characteristics that describe these clusters. In comparison, when only using subspace clustering without any background knowledge, the expert would either obtain a plethora of irrelevant clusters (e.g. containing participants with the same shoes size, even if the shoe size is not associated with the concept) or the algorithm would not terminate at all due to the high combinatorial complexity of the search space that many subspace clustering methods have to deal with. Authors point out that if constraints are beneficial to a plain clusterer, they might be even more useful for high-dimensional data in guiding a subspace clusterer by restricting the search space and presenting the user results / clusters that are actually relevant to their concept [87].

Two dedicated subspace clustering algorithms that exploit constraints are Constrained K-Means [88] and SMVC [89]. Constrained K-Means uses instance-level constraints to assign an instance x to the closest cluster so that the assignment decision does not lead to a violation of the constraints related to x. SMVC models multiple clusterings and related dimensions (views) of the data with the help of a Bayesian framework. Because views define different, alternative clusterings of the data, the algorithm does not necessarily drop views that violate many constraints. These methods exhibit some undesired characteristics w.r.t. real-world application settings, e.g. prior assumptions about number and shape of clusters within subspaces or the focus on different, alternative clusterings. Contrary, authors in [87] present a general framework, SC-MINER, that can be utilized to employ basic grid- and distance-based subspace clusterers in constraint-based subspace clustering by pushing and checking constraints during the enumeration process of cluster candidates. SC-MINER automatically ensures that objects under Must-Link constraint are put into the same cluster and avoids the generation of candidate clusters that contain objects under Not-Link constraints, thus reducing the search space and improving the quality of results. However, SC-MINER requires a subspace-independent parameter setting which is a density-threshold (clusters are only dense if they exceed the threshold) that does not consider that less dense regions in a high-dimensional subspace could still contain useful knowledge in comparison to slightly denser subspaces of lower cardinality. The algorithm thus avoids to handle the natural decrease in density that comes with higher cardinality subspaces, omitting potentially relevant clusters in the process.

4. EVO-Extract: Extracting Evolution Features

The first main component of our framework is EVO-Extract. EVO-Extract deals with the task of deriving relevant knowledge from an object's evolution as motivated in Section 1.1.2. In this chapter we address the following research questions:

RQ 1: How can we codify the implicit dimensionality of an object's historical records to improve object classification?

RQ 1.1: How to identify relevant feature-value sequence patterns of evolving objects for classification while preserving relevant information?

RQ 1.2: How to model the evolution of complex objects for classification?

EVO-Extract expects a dataset D of evolving objects to be applied onto, i.e. a dataset with a non-empty time-series feature set G. EVO-Extract uses the historical records, given by the set of time-series features G, and ground-truth, given by the subset of training data $D_{tr} \subseteq D$ where the class is known for each instance $x \in D_{tr}$, to extract different types of Evolution Features.

This chapter begins with a description of the preprocessing component that transforms datasets of evolving objects into the scheme required by certain EVO-Extract and STATIC-Select sub-components, and the problem definition of EVO-Extract. In Section 4.3 we show how features based on sequential feature value patterns (patterns in the evolution of objects) can be derived. Section 4.4 describes how pre-defined features based on individual and cluster evolution can be codified. Section 4.5 shows the evaluation of our methods with the help of classification workflows that use our derived Evolution Features. At last, Section 4.6 concludes this chapter and provides answers to the research questions. Parts of Section 4.1 - Section 4.6 are from our previous publications [27–29] (with modifications).

4.1 Preprocessing

The methods to derive object- and group-based Evolution Features in the EVO-Extract component, as well as the complete STATIC-Select component, expect that the time-series features of an evolving dataset are transformed into a single set of static features. To do this, the preprocessing component treats every feature observation $g_{i,t}$ of a time-series feature g_i as a single static feature and inserts it into a set G_{stat}, i.e. given T time-points, $G_{stat} = G_{\cdot,0} \cup G_{\cdot,1} \cup ... \cup G_{\cdot,T-1}$. Doing this allows the framework to apply traditional feature selection methods on the observations of time-series features to find sets of important observations. Additionally, the preprocessing component has the task to conduct a normalization of all static features in F and G_{stat} into range $[0; 1]$ since distances in different feature-(observation) subspaces must be comparable (EVO-Extract and STATIC-Select construct and select features based on position and distances of objects in different feature subspaces). For each continuous feature $f \in F \cup G_{stat}$ the preprocessing component assigns each instance $x \in D$ a new feature-value $f_{new}(x)$ as follows:

$$f_{new}(x) = \frac{f(x) - f_{min}}{f_{max} - f_{min}}, \tag{4.1}$$

$$\text{with } f_{max} = max_{x \in D}(f(x)), \tag{4.2}$$

$$\text{and } f_{min} = min_{x \in D}(f(x)), \tag{4.3}$$

where f_{min} is the minimal and f_{max} is the maximal feature-value observed in the respective dataset for feature f. The new values $f_{new}()$ then replace the original ones. For interpretability reasons the normalization model may be stored and used later to restore the original values.

Further, we eliminate features that satisfy both of the following two conditions when used with STATIC-Select (second main component): (1) a frequency-ratio (ratio between the most frequent value to the second most frequent value of a feature) of at least 95% and (2) with a percentage of distinctive values below the observed number of distinct values for the feature, divided by the number of all observed values for it.

4.2 Problem Definition

Let $model(D_{tr}, F)$ be a classification model learnt on training data $D_{tr} \subseteq D$ and feature set F, and let $perf(model, D_{te})$ be the classification model performance that is obtained when using classification model *model* to classify each instance $x \in D_{te} \subseteq D$ (such as Accuracy, Specificity etc., we present the utilized evaluation measures as well as an aggregated "rank" of these measures that may be used as a decision basis in Section 4.5.2). Let us further denote $F^*, G*, G_{evo}*$ as a non-empty subset of relevant features (towards the class variable) of F, G, G_{evo}.

The goal of EVO-Extract is to use feature sets G, G_{stat} and dataset $D = D_{tr} \cup D_{te}$ to create Evolution Features set G_{evo}, i.e. $EVO{-}Extract(G, D) = G_{evo}$, so that

$$perf(model(D_{tr}, F^* \cup G^* \cup G^*_{evo}), D_{te}) > perf(model(D_{tr}, F^* \cup G^*), D_{te}) \tag{4.4}$$

holds true.

4.3 Deriving Pattern-based Evolution Features

The first sub-component of EVO-Extract summarizes the values of each time-series feature, exploiting similar numerical sequences of different instances to derive new summary features, so called pattern-based Evolution

Features. EVO-Extract aims to automatically find relevant groups of similar sequences in a data-driven way: Objects that exhibit similar sequences in a single time-series feature are grouped together according to a base clusterer. EVO-Extract creates a number of alternative clustering for those sequences. Then, for each time-series feature EVO-Extract creates exactly one new Evolution Feature by codifying the clustering with the highest information towards the target variable. Here, each instance is assigned the cluster membership of its corresponding sequence as the Evolution Feature value. This leads to the creation of features where instances with similar sequences share the same feature-value in the respective Evolution Feature, whereas the values themselves are derived from clusterings that show evidence of containing information towards the target concept. Figure 4.1 depicts the basic workflow for the creation of these features.

EVO-Extract first creates sets of sequential values for each time-series feature and then clusters each set a number of times (cf. Algorithm 1) to find groups of similar sequences. In the following we introduce the notion of *sequence-examples* and *sequence-sets*:

Sequence-example: sequence-examples contain the sequential values of one instance for a single time-series feature. Let G be the set of time-series features and D the complete dataset. A sequence-example of an instance $x \in D$ and time-series feature g_i with $|G_{i,\cdot}| = n$ observations is the vector $s(x, g_i) = (g_{i,0}(x), ..., g_{i,n}(x))$ which represents a point in the $|G_{i,\cdot}|$-dimensional feature observation space. For example, if $|G_{i,\cdot}| = 7$ and g_i is the time-series feature describing the Body-Mass-Index measurement outcome, then for an instance x, $s(x, g_i)$ would contain a sequence of seven BMI measurement outcomes.

Sequence-set: sequence-sets contain all sequence-examples of a single time-series feature. For a time-series feature g_i the corresponding sequence-set is defined as $S(g_i, D) = \{s(x, g_i) | x \in D\}$.

To identify sets of similar sequences, we introduce a notion of distance between sequence-examples based on the projection of instances onto the feature observation space induced by a sequence-set.

Distance between sequence-examples: Let $d()$ denote any distance function with $d(x_1, x_2)$ being the distance between instance x_1 and x_2 in the complete feature space of dataset D. Given a time-series feature g_1, we de-

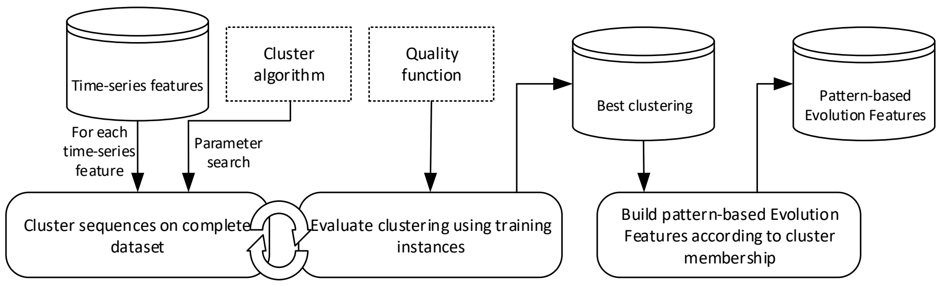

Time-series features
Cluster algorithm
Quality function
Best clustering
Pattern-based Evolution Features
For each time-series feature
Parameter search
Cluster sequences on complete dataset
Evaluate clustering using training instances
Build pattern-based Evolution Features according to cluster membership

Figure 4.1: Basic workflow and involved components to create pattern-based Evolution Features. Dashed rectangles describe general component families where the specific algorithm choice is up to the user. ©ACM 2018. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in [28], http://dx.doi.org/10.1145/3167132.3167162

fine the distance $d()$ between the sequence-examples $s(x_1, g_i)$ and $s(x_2, g_i)$ as $d(s(x_1, g_i), s(x_2, g_i)) = d(\pi_{G_{i,\cdot}}(x_1), \pi_{G_{i,\cdot}}(x_2))$. The distance between sequence examples is calculated based on the HEOM as described in Section 2.1.3, adjusted for the instance projections given by the sequences:

$$d(\pi_{G_{i,\cdot}}(x_1), \pi_{G_{i,\cdot}}(x_2)) = \sqrt{\sum_{g_{i,t} \in G_{i,\cdot}} \delta(g_{i,t}(x_1), g_{i,t}(x_2))^2} \tag{4.5}$$

EVO-Extract builds one set of sequence-examples for each time-series feature and investigates them independently. Therefore, the number of observations can differ for each time-series feature without impacting the component. Also, if missing values exist, they must not be imputed since our distance measure deals with them.

4.3.1 Creating Alternative Clusterings

EVO-Extract uses the density-based clustering algorithm DBSCAN [48]. DBSCAN exhibits a number of advantages compared to alternative methods as described in Section 3.1. One of the main advantages for our application is that identified clusters are not limited to linear boundaries so that no assumptions on the underlying features distributions are made. For each time-series feature g_i in dataset D, EVO-Extract invokes DBSCAN on sequence-set $S(g_i, D)$ an $altNum+1$ times according to different parameter settings. For these alternative clusterings the $MinPts$ parameter is fixed to $MinPts = round(ln(|D|))$ according to the suggestion of Ester et al. [48]. This ensures that all found clusters do not consist of singletons, but instead have a lower limit to their size that depends on the size of the complete dataset.

Then, given the fixed $MinPts$, EVO-Extract automatically calculates a first "reasonable" Eps-parameter value, according to the $kDist$, $kDistList$ and the *knee-point method* [29]. These notions are defined as follows:

kDist: Let $nn(s(x, g_i), k)$ denote the k-nearest neighbor of sequence-example $s(x, g_i)$ in the sequence space induced by g_i, i.e. the sequence-example with the k-th lowest distance $d()$ to $s(x, g_i)$. The $kDist(s(x, g_i))$ is defined as the distance $d(s(x, g_i), nn(s(x, g_i), k))$.

kDistList: Let $S(g_i, D)$ be a sequence-set with $n = |S(g_i, D)|$. Then, we define the $kDistList(S(g_i, D))$ as the ordered vector of $kDist$-values $(kDist_1, ..., kDist_n)$ induced by $S(g_i, D)$ with the first vector-value

$$kDist_1 = max_{s \in S(g_i,D)} \{kDist(s)\},$$

the last vector-value

$$kDist_n = min_{s \in S(g_i,D)} \{kDist(s)\},$$

and where $\forall i' \in \{2, ..., n\} : kDist_{i'} \leq kDist_{i'-1}$ holds true.

Knee-point method: Let $k = MinPts$, $S(g_i, D)$ be the sequence-set used for clustering and $n = |S(g_i, D)|$ the size of the sequence-set. First, the $kDistList(S(g_i, D))$ is computed. Each $kDist \in kDistList$ defines the required Eps-value so that the corresponding sequence-example is a core point. Then, an empty two-dimensional real-valued space is created and one datapoint per sequence-example inserted, i.e. $\forall i' \in \{1, ..., n\}$ the algorithm inserts the point $z_{i'} = (i' - 1, kDist_{i'})$. The resulting graph is denoted as the kDist graph. After that, a line $line(z_1, z_n)$ between the points $z_1 = (0, kDist_1)$ and $z_n = (n-1, kDist_n)$ in the kDist graph is computed (the points associated with the first list element and last element of $kDistList$). Following this, $\forall i' \in \{1, ..., n\}$ the shortest euclidean distance, $Euclid(line(z_1, z_n), z_{i'})$, between $line(z_1, z_n)$ and $z_{i'}$ is calculated. Finally, the Eps-value is set to $Eps = max_{i' \in \{1,...,n\}} \{kDist_{i'}\}$, correspondent to the sequence-example with the highest $Euclid(line(z_1, z_n), z_{i'})$ (knee-point).

EVO-Extract chooses as the initial Eps-parameter setting the $kDist \in kDistList$ value that belongs to the sequence-example that maximizes the shortest distance between $line(z_1, z_n)$ and its $kDist$ graph point. This chosen sequence-example is correspondent to a knee-point in the $kDist$ graph. For the given $MinPts$, this heuristic enables DBSCAN to choose a knee-point in the k-dist graph that preserves a number of dense regions as clusters. Figure 4.2 shows an example of a $kDist$ graph, knee-point and associated Eps-value and clustering.

In addition to this first "best bet" parameter setting, EVO-Extract conducts an $altNum$ number of alternative clusterings, as shown in Algorithm 1, according to the distribution of the distances between the points in the kDist graph and the calculated line. For each of these clusterings a random $kDist$-value is chosen for the Eps parameter. Since high distances between the $line$ and points in the kDist graph are more favorable than low distances (high distances are correspondent to knee-points), the probabilities for choosing a specific Eps value are not equal but weighted according to the $Euclid(line(z_1, z_n), z)$ distance, with higher distances leading to a higher probability. For time-series feature g_i, the probability to choose a specific $kDist_{i'} \in kDistList$ as Eps-value, is the fraction of $z_{i'}$'s distance to $line$ w.r.t. the sum of distances of all points of the $kDist$ graph to $line$:

$$p(kDist_{i'}) = \frac{Euclid(line(z_1, z_n), z_{i'})}{\sum_{j=1}^{n} Euclid(line(z_1, z_n), z_j)}. \tag{4.6}$$

4.3.2 Cluster Evaluation

Given a time-series feature g_i, EVO-Extract evaluates each clustering that was created according to Section 4.3.1 independently with the help of our adjusted Gain-Ratio notion. Let $\mathcal{C}_{g_i,D}$ denote a clustering (set of clusters) of

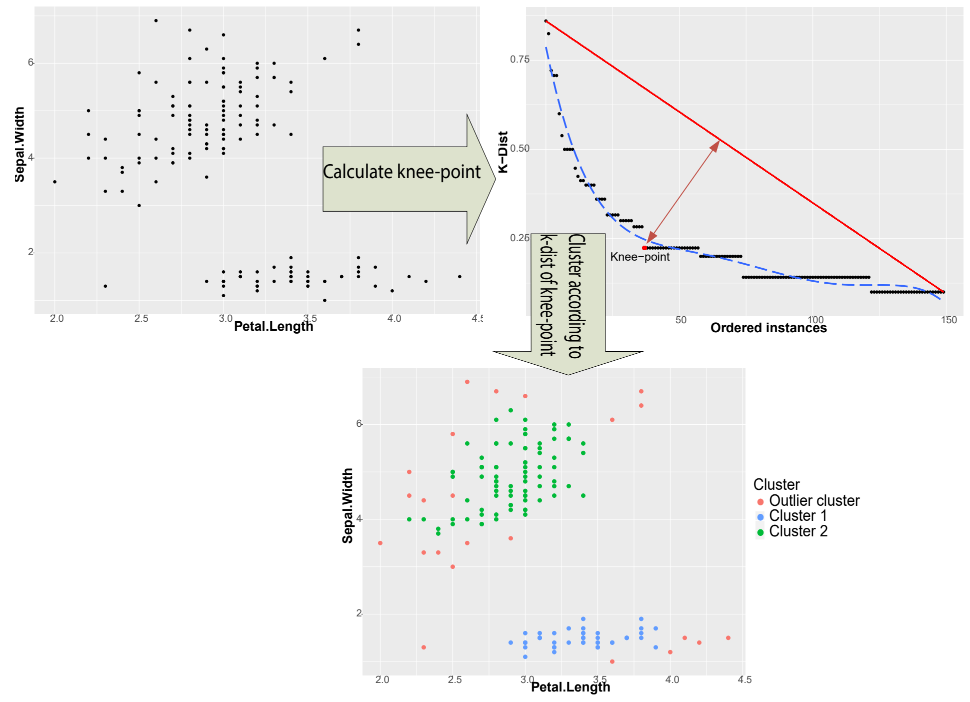


Figure 4.2: Basic workflow to generate an *Eps*-value according to the knee-point method. Left plot: Scatterplot of a two-dimensional projection of the Iris data. Right plot: Based on the 5-nearest neighbor (5-nn) distances ($MinPts = 5$), we compute a line between the object associated with the lowest 5-nn distance and the object with the highest 5-nn distance. EVO-Extract sets the *Eps*-parameter to the 5-nn distance of the object (marked as a red dot) with the highest distance to the line, which marks a knee-point. Here, all instances to the right of the marked object will be part of a cluster (because their 5-nn distance is lower than the chosen *Eps* value) and all objects to the left will be marked as outliers when using DBSCAN. Lower Plot: DBSCAN clustering with the derived parameter settings on the IRIS data, colors mark cluster memberships.

```
**Data:** Sequence-set $S(g_i, D)$, number of alternative clusterings
        $altNum$
**Result:** Set of alternative clusterings $\mathcal{CL}$
$kDistList \leftarrow \emptyset$ ; // Initialize empty $kDistList$
$\mathcal{CL} \leftarrow \emptyset$ ; // Initialize empty set of clusterings
$MinPts \leftarrow round(ln(|S(g_i, D)|))$; // Calculate MinPts value
$k \leftarrow MinPts$; // Set $k$
for $s \in S(g_i, D)$ do
    $kDistList \leftarrow kDistList \cup kDist(s)$; // Add to the list
     the distance of $s$ to its $k$-nearest neighbor.
end
$sortDescending(kDistList)$ ; // Sort list (descending)
$line \leftarrow calculateLine(0, kDistList[0], |S(g_i, D)| -$
 $1, kDistList[|S(g_i, D)| - 1])$ ; // Create the line between
 first and last element.
$dist_{max} \leftarrow -1$ ; // Initialize current highest distance
$Eps \leftarrow -1$ ; // Initialize the knee-point MinPtsDist
for $j = 1 : |S(g_i, D)|$ do
    if $dist(line, j - 1, kDistList(j - 1)) > dist_{max}$ then
        $dist_{max} \leftarrow dist(line, j - 1, kDistList(j - 1))$ ; // Update
         highest observed distance
        $Eps \leftarrow kDistList(j - 1)$ ; // Update knee-point $k$Dist
    end
end
$\mathcal{CL} \leftarrow \mathcal{CL} \cup DBSCAN(MinPts, Eps)$ ; // Conduct and store
 clustering according to knee-point $Eps$ value
$probVector \leftarrow createPobVector(S(g_i, D))$; // Create the
 probability vector where each entry $probVector[o]$ with
 $o \in \{1, ..., |S(g_i, D)|\}$ is given by $p(kDist_o)$ as shown in
 Equation 4.6
for $i = 1 : altNum$ do
    $Eps \leftarrow chooseRandomDist(probVector, kDistList)$ ;
     // Choose a random $Eps$-value from the $k$-distance
     list where the chance of choosing element
     $kDistList[j]$ is given by $probVector[j]$
    $\mathcal{CL} \leftarrow \mathcal{CL} \cup DBSCAN(MinPts, eps)$ ; // Conduct and
     store clustering according to chosen $Eps$ value
end
```

Algorithm 1: Creating alternative clusterings.

the sequence-example set $S(g_i, D)$ and $D_{tr}(C) = \{x \in D_{tr} | s(x, g_i) \in C\}$ denote the set of training instances for which it holds that their corresponding sequence-examples $s()$ are present in cluster $C \in \mathcal{C}_{g_i,D}$. Also, let $D_{tr} \subseteq D$ be the complete set of training instances with disclosed class label (equals the union of all $D_{tr}(C) \subseteq D_{tr}$). Let us further denote Y as the set of class labels and $\mathcal{D}_{tr}(\mathcal{C}_{g_i,D}) = \{D_{tr}(C) | C \in \mathcal{C}_{g_i,D}\}$ as the superset of training instance partitions induced by the clusters of $\mathcal{C}_{g_i,D}$. The Gain-Ratio of the clustering is then calculated as

$$GR(Y, \mathcal{D}_{tr}(\mathcal{C}_{g_i,D})) = \frac{IG(Y, \mathcal{D}_{tr}(\mathcal{C}_{g_i,D}))}{H(\mathcal{D}_{tr}(\mathcal{C}_{g_i,D}))} \tag{4.7}$$

where

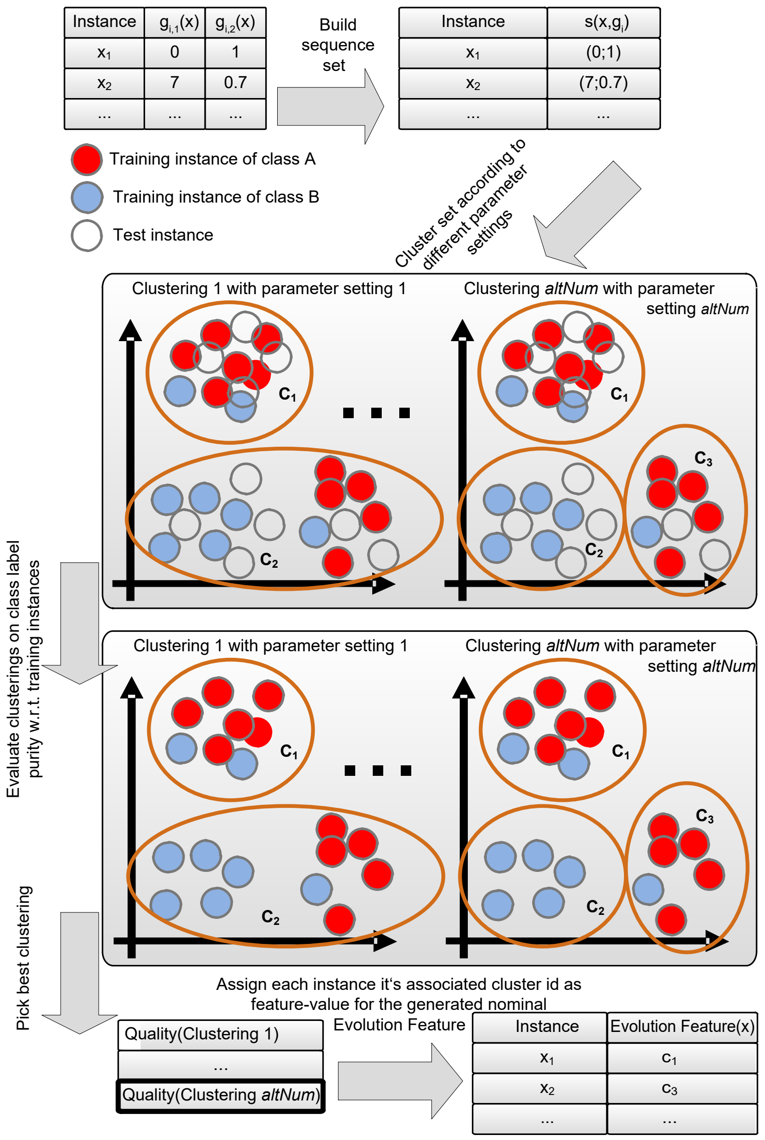
$$\begin{aligned} IG(Y, \mathcal{D}_{tr}(\mathcal{C}_{g_i,D})) &= H(Y, D_{tr}) - H(Y, \mathcal{D}_{tr}(\mathcal{C}_{g_i,D})), \\ H(Y, D_{tr}) &= -\sum_{y \in Y} p(y, D_{tr}) * log_2(p(y, D_{tr})), \\ H(Y, \mathcal{D}_{tr}(\mathcal{C}_{g_i,D})) &= \sum_{I \in \mathcal{D}_{tr}(\mathcal{C}_{g_i,D})} |I|/|D_{tr}| * H(Y, I), \\ H(\mathcal{D}_{tr}(\mathcal{C}_{g_i,D})) &= -\sum_{I \in \mathcal{D}_{tr}(\mathcal{C}_{g_i,D})} |I|/|D_{tr}| * log_2(|I|/|D_{tr}|) \ . \end{aligned}$$

Note that clusters can contain instances with and without disclosed label information (clustering is conducted on all instances). Therefore, the clustering evaluation considers only a cluster's instances that are part of the training dataset. Once the best clustering for a feature g_i is found, a new nominal Evolution Feature $evo\_pattern_{g_i}$ is constructed (and inserted into F), where each $x \in D$ is assigned the cluster-membership id of its associated sequence-example $s(x, g_i)$. Algorithm 2 shows the complete procedure to derive pattern-based Evolution Features.

The workflow to derive pattern-based Evolution Features for an example feature with two observations is depicted in Figure 4.3.

```
Data: Dataset $D$, training dataset $D_{tr} \subseteq D$, labels $Y$, time-series
      features $G$, static features $F$, number of alternative
      clusterings $altNum$
Result: Set of updated static features $F$
for each $g_i \in G$ do
    $GR_{opt} \leftarrow -1$; // Initialize Gain-Ratio
    $\mathcal{C}_{opt} \leftarrow \emptyset$; // Initialize clustering variable associated
      with highest Gain Ratio
    $S(g_i, D) \leftarrow buildSequenceSet(g_i, D)$;
    $\mathcal{CL} \leftarrow createClusterings(S(g_i, D), altNum)$; // Conduct
      alternative clusterings and store them according to
      Section 4.3.1
    for each $\mathcal{C}_{g_i,D} \in \mathcal{CL}$ do
        $\mathcal{D}_{tr}(\mathcal{C}_{g_i,D}) \leftarrow \emptyset$; // Initialize superset of training
          set partitions
        for each $C \in \mathcal{C}_{g_i,D}$ do
            $D_{part} \leftarrow C \cap \pi_{G_{i,\cdot}}(D_{tr}))$;
            $\mathcal{D}_{tr}(\mathcal{C}_{g_i,D}) \leftarrow \mathcal{D}_{tr}(\mathcal{C}_{g_i,D}) \cup \{D_{part}\}$;
        end
        // Create set of training dataset partitions
           induced by $\mathcal{C}_{g_i,D}$. A training dataset partition
           induced by $C$ is the intersection between $C$
           and the projection $\pi_{G_{i,\cdot}}(D_{tr})$, i.e. the set of
           sequence-examples in $C$ that come from
           instances which are part of $D_{tr}$.
        if $GR(Y, \mathcal{D}_{tr}(\mathcal{C}_{g_i,D})) > GR_{opt}$ then
            $GR_{opt} \leftarrow GR(Y, \mathcal{D}_{tr}(\mathcal{C}_{g_i,D}))$;
            $\mathcal{C}_{opt} \leftarrow \mathcal{C}_{g_i,D}$;
        end
    end
    // Find clustering with highest Gain Ratio
    $F \leftarrow F \cup \{evo\_pattern_{g_i}\}$;
    // Create Evolution Feature out of best clustering
    for each $x \in D$ do
        $evo\_pattern_{g_i}(x) \leftarrow getClusterID(s(x, g_i), \mathcal{C}_{opt})$;
    end
    // Assign each instance the cluster membership id in
       $\mathcal{C}_{opt}$ of its sequence-example $s(x, g_i)$ as feature
       value
end
```

Algorithm 2: Creating pattern-based Evolution Features.



Instance	$g_{i,1}(x)$	$g_{i,2}(x)$
x_1	0	1
x_2	7	0.7
...	...	...

Instance	$s(x,g_i)$
x_1	(0;1)
x_2	(7;0.7)
...	...

Quality(Clustering 1)
...
Quality(Clustering *altNum*)

Instance	Evolution Feature(x)
x_1	c_1
x_2	c_3
...	...

Build sequence set
● Training instance of class A
● Training instance of class B
○ Test instance
Cluster set according to different parameter settings
Clustering 1 with parameter setting 1
Clustering *altNum* with parameter setting *altNum*
C_1
C_2
C_3
Evaluate clusterings on class label purity w.r.t. training instances
Pick best clustering
Assign each instance it's associated cluster id as feature-value for the generated nominal Evolution Feature

Figure 4.3: Example workflow for the construction of pattern-based Evolution Features from a time-series feature g_i with two observations using DBSCAN. First, the sequence-set is build. Then, $altNum$+1 clusterings are conducted: One knee-point based clustering and $altNum$ clusterings with random kDist choices for the *eps*-parameter (probabilities of choosing a kDist value are weighted with the respective objects's distance to *line* as described in Section 4.3.1). At last, the best clustering according to the Gain-Ratio is chosen and an Evolution Feature is created where each instance is assigned the cluster membership of its sequence-example. ©ACM 2018. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in [28], http://dx.doi.org/10.1145/3167132.3167162

4.4 Deriving Object- & Group-based Evolution Features

The remaining sub-components of EVO-Extract exploit ground-truth that is provided by labeled training data and the individual and group-based evolution of objects, from one time-point to the next, to create Evolution Features. Instead of settling for the given feature-vectors in a dataset, EVO-Extract tracks differences of instances in their feature-values across time-points and codifies them within new features. For each time-series feature $g_i \in G$ with feature observation set $G_{i,\cdot}$, EVO-Extract builds each pair of distinct feature-observations $g_{i,t}, g_{i,t'} \in G_{i,\cdot}$ with $g_{i,t} \neq g_{i,t'}$ and calculates the real $real\_d(g_{i,t}(x), g_{i,t'}(x))$, absolute $absolute\_d(g_{i,t}(x), g_{i,t'}(x))$ and relative $relative\_d(g_{i,t}(x), g_{i,t'}(x))$ differences between the respective feature observation values for each instance $x \in D$ as follows (analogue to the HEOM in Section 2.1.3):

$$real\_d(g_{i,t}(x), g_{i,t'}(x)) = \delta(g_{i,t}(x), g_{i,t'}(x)), \tag{4.8}$$

$$absolute\_d(g_{i,t}(x), g_{i,t'}(x)) = |\delta(g_{i,t}(x), g_{i,t'}(x))|, \tag{4.9}$$

$$relative\_d(g_{i,t}(x), g_{i,t'}(x)) = \frac{\delta(g_{i,t'}(x), g_{i,t}(x))}{g_{i,t}(x)} * 100. \tag{4.10}$$

For each distinct pair of feature observations $g_{i,t}, g_{i,t'}$ a new Evolution Feature $evo\_real_{g_{i,t},g_{i,t'}}$,$evo\_absolute_{g_{i,t},g_{i,t'}}$ and $evo\_relative_{g_{i,t},g_{i,t'}}$ is constructed, where each instance is assigned the respective difference as the feature-value, for example $evo\_absolute_{g_{i,t},g_{i,t'}}(x) = absolute\_d(g_{i,t}(x), g_{i,t'}(x))$.

Additionally, EVO-Extract not only considers the evolution of each instance's feature vector, but also the evolution of the instance in relation to its neighbors. Here, Evolution Features are derived based on the clustering of instances within informative subspaces and the relation of the instances w.r.t. its peers and clusters. To derive such a set of Evolution Features, EVO-Extract expects a labeled training dataset (for example the labeled part of the dataset which may be later used for classification model learning). On this training set, the module applies Correlation-based Feature Selection [12] (using feature observation set G_{stat}) to identify a subset $G^*_{stat} \subseteq G_{stat}$ consisting of relevant feature observations associated with the target variable. To ensure that clusters can be tracked across time-points and results are comparable, the module forces that the feature observation subspaces are associated with exactly the same time-series features in each moment by creating a set G^{tk} that contains all feature observations $g_{i,t}$ of all time-series features which are observed in each moment t and contribute to G^{*}_{stat}, i.e.

$$G^{tk} = \{g_{i,t} | \forall t = 0, ..., T : g_{i,t} \in G_{i,\cdot} \wedge \exists t \in \{0, ..., T\} : g_{i,t} \in G^{*}_{stat}\}.$$

We denote $G^{tk}_t \subseteq G^{tk}$ as the subset of features of G^{tk} associated with the feature observations of a specific time-point t (the feature subset used for $t$rac$k$ing the instance positions across time-points). After building G^{tk}, the module conducts one clustering on each projection $\pi_{G^{tk}_t}(D)$, i.e. one clustering in each moment t on D, using the subspace induced by G^{tk}_t. We denote the cluster that contains instance x in the clustering of projection $\pi_{G^{tk}_t}(D)$ as $C(\pi_{G^{tk}_t}(x))$ and the neighborhood that contains the k-nearest neighbors of x in G^{tk}_t as $NN(\pi_{G^{tk}_t}(x), k)$.

Since some of the object- and group-based Evolution Features are based on special objects and measures within clusters, we define those notions in the following:

Centroid: Let C be a cluster in feature space $F = \{f_1, f_2, ..., f_n\}$. We denote the centroid of C as the "best" representative of a spherical cluster, i.e. the mean feature-vector

$$cent(C) = (f_1(cent(C)), f_2(cent(C)), ..., f_n(cent(C)))$$

of all contained cluster members with feature-values

$$f_i(cent(C)) = \frac{\sum_{x \in C} f_i(x)}{|C|}, \forall f_i \in F[43].$$

Medoid: Let C be a cluster in feature space F and $d()$ a distance function. The medoid $med(C)$ of a cluster C is the cluster-member $x \in C$ for which the sum of all distances to its cluster peers is minimal, i.e.

$$med(C) = \operatorname*{argmin}_{x \in C} \left(\sum_{x' \in C} d(\pi_F(x), \pi_F(x')) \right) [90].$$

Shortest path length of density-reachable objects: Let p, q be two instances and $\mathcal{CH}$ denote the superset of all possible chains of objects $(x_1, ..., x_n)$ so that $x_1 = p, x_n = q$ and $\forall i = 1, ..., n-1 : x_{i+1}$ is *directly density-reachable* from x_i holds true. The shortest path length of density-reachable objects from x_1 to x_2 is then defined as $path(x_1, x_2) = \min_{CH \in \mathcal{CH}}(|CH|)$.

Local Outlier Factor: Let $d()$ denote a distance function, $MinPtsDist(\pi_F(x))$ the distance from x to its $MinPts$-nearest neighbor in feature space F, and

$$N_{MinPtsDist}(\pi_F(x)) = \left\{ x' | d(\pi_F(x), \pi_F(x')) \leq MinPtsDist(\pi_F(x)) \right\}$$

the set of instances in the $MinPtsDist$ neighborhood of x in F. The Local Outlier Factor of an instance x in F is the average ratio of local reachability of x from its neighbors $x' \in N_{MinPtsDist}(\pi_F(x))$ to the average reachability of all neighbors x'. It is defined as [91]:

$$LOF_{MinPts}(\pi_F(x)) = \frac{\sum_{x' \in N_{MinPtsDist}(\pi_F(x))} \frac{lrd_{MinPts}(\pi_F(x'))}{lrd_{MinPts}(\pi_F(x))}}{|N_{MinPtsDist}(\pi_F(x))|}, \tag{4.11}$$

where

$$lrd_{MinPts}(\pi_F(x)) = 1/ \left(\frac{\sum_{x' \in N_{MinPtsDist}(\pi_F(x))} reachDist_{MinPts}(\pi_F(x), \pi_F(x'))}{|N_{MinPtsDist}(\pi_F(x))|} \right),$$
$$reachDist_{MinPts}(\pi_F(x), \pi_F(x')) = max \left\{ MinPtsDist(\pi_F(x')), d(\pi_F(x), \pi_F(x')) \right\}.$$

Silhouette coefficient: Let $d()$ denote a distance function and $\mathcal{CL}$ a clustering (set of clusters) in feature space F. The Silhouette coefficient for an instance x is a ratio between how well an instance is separated from instances of other clusters, and its cohesion w.r.t. to its cluster peers. It is defined as [92]:

$$silh(\pi_F(x)) = \frac{sep(\pi_F(x)) - coh(\pi_F(x))}{max\left\{ sep(\pi_F(x)), coh(\pi_F(x)) \right\}}, \tag{4.12}$$

where

$$sep(\pi_F(x)) = \frac{1}{|\mathcal{CL}| - 1} \sum_{C' \in \mathcal{CL} \setminus C(\pi_F(x))} \frac{1}{|C'|} \left(\sum_{x' \in C'} d(\pi_F(x), \pi_F(x')) \right),$$

$$coh(\pi_F(x)) = \frac{1}{|C(\pi_F(x))| - 1} \sum_{x' \in C(\pi_F(x)) \setminus \{x\}} d(\pi_F(x), \pi_F(x')).$$

Given the clustering results (parameters according to the knee-point method, i.e. Section 4.3.1) and the neighborhood definition used by the clusterer (with $MinPts$-value according to the heuristic described in Section 4.3.1), the Evolution Features as shown in Section 4.4 are constructed and inserted into F.

4.5 Evaluation

In this section, EVO-Extract is evaluated as follows: Given a classification problem w.r.t. evolving objects, how do classifier perform when using the Evolution Features returned by EVO-Extract in conjunction with the original feature set? To answer this question we evaluate an extensive number of state-of-the-art classification algorithms w.r.t. their classification performance on real-world datasets.

In the next subsections we first describe the datasets we use and give an overview about the variants under evaluation. In Section 4.5.3 we show and discuss the results of our experiments. Finally, we draw conclusions w.r.t. EVO-Extract in Section 4.6.

4.5.1 Datasets

The first dataset originates from the longitudinal, epidemiological "Study of Health in Pomerania (SHIP)" [93] of two cohorts in north-east Pomerania. Approximately every four years, the participants of SHIP underwent an extensive recurring examination program that encompasses interviews, exercise tests, laboratory analysis, ultrasound examinations and magnetic resonance tomography (MRT). Currently data for the three study waves SHIP-0, SHIP-1 and SHIP-2 exists. The first examinations of SHIP-0 (n=4308) were performed between 1997 and 2001. The followups SHIP-1 (n=3300)

Evolution Features **for each x in each moment t of G^{tk}**	
`cluster_`t	ID of $C(\pi_{G_t^{tk}}(x))$, the cluster to which x belongs at moment t in G^{tk}
`dist_to_cent_`t	distance of x to $cent(C(\pi_{G_t^{tk}}(x)))$
`dist_to_med_`t	distance of x to $med(C(\pi_{G_t^{tk}}(x)))$
`is_rep_`t	1, if $x = med(C(\pi_{G_t^{tk}}(x)))$, else 0.
`path_to_rep_`t	length of the chain from cluster medoid to x $(path(med(C(\pi_{G_t^{tk}}(x))), x))$
`lof_`t	local outlier factor value $LOF_{MinPts}(\pi_{G_t^{tk}}(x))$
`frac_`$class$`_in_NN_`t	fraction of the instances within the ϵ neighborhood of x at t in G^{tk} that belong to class $class$; one feature per class is derived
Evolution Features **for each x in each pair t, t' of G^{tk}**	
`silhouette_`t	$silh(\pi_{G_t^{tk}}(x))$ for the clustering at t in G^{tk}
`frac_peers_`t`_`t'	fraction of instances that are cluster peers of x both in t and t' of G^{tk}, i.e. $\frac{\left\|\left\{C(\pi_{G_t^{tk}}(x))\cap C(\pi_{G_{t'}^{tk}}(x))\setminus\{x\}\right\}\right\|}{\max\|C(\pi_{G_t^{tk}}(x))\|,\|C(\pi_{G_{t'}^{tk}}(x))\|}$
`frac_MinPts_NN_`t`_`t'	fraction of instances who are among the $MinPts$-nearest neighbors in both t and t', i.e. $\frac{\left\|\left\{NN(\pi_{G_t^{tk}}(x),MinPts)\cap NN(\pi_{G_{t'}^{tk}}(x),MinPts)\right\}\right\|}{MinPts}$
`real_diff_`a`_`t`_`t'	difference $a(\pi_{G_t^{tk}}(x)) - a(\pi_{G_{t'}^{tk}}(x))$, where $a \in \{silh, LOF_{MinPts}\}$
`stays_outlier_`t`_`t'	1, if x is outlier in both t and t' of G^{tk}, else 0
`becomes_outlier_`t`_`t'	1, if x is not an outlier in t and is an outlier in t' of G^{tk}, else 0
`was_outlier_`t`_`t'	1, if x is an outlier in t and is not an outlier in t' of G^{tk}, else 0
`never_outlier_`t`_`t'	1, if x is not an outlier in both t and t' of G^{tk}, else 0

Table 4.1: Generated object- and group-based Evolution Features to be associated with each instance x: features on changes of clusters and in the instance's position (in the feature observation space) relative to the cluster and to its closest neighbors are constructed. Note that for the binary features that encode outlier status we use the DBSCAN assignment from the respective clusterings.

and SHIP-2 (n=2333) were undertaken between 2002-2006 and 2008-2012. For SHIP, we received data to investigate the liver disorder hepatic steatosis, that is also known as fatty liver which is present in approximately 30% of all adults [93], for a subset of 886 study participants (hereafter denoted as SHIP-886) which participated in all three waves. Although not harmful per se, possible followup diseases like steatohepatitis and liver cirrhosis can cause severe harm [94]. The class variable is derived from a liver-fat measurement of a MRT examination that was only conducted during SHIP-2: participants with a liver fat concentration value of less than 5% are assigned the low class (class 1), participants with liver-fat concentrations in the range $(5\%; 10\%]$ are assigned the intermediate class (class 2) and participants with greater values are assigned the high class (class 3). The complete SHIP dataset is further split into two subsets consisting of the female and male participants. Both subsets are investigated individually.

The second dataset we use comes again from a longitudinal, epidemiological study. Here, a population from Denmark is investigated in the WHO project MONICA (Multinational MONItoring of trends and determinants in CArdiovascular disease). MONICA focuses on risk factors of cardio-vascular diseases and causes / trends w.r.t. differences in the mortality between countries [95]. In this study, 37 centers from 21 countries participated in a time-frame between 1976 and 2002. Over 30,000 residents from the south-west of Copenhagen County with an age between 25 and 74 years, participated in three waves from 1982 to 1991 [96] in MONICA. Baseline examinations were conducted in 1982-1986 (DAN-MONICA I), the first follow-up in 1986-1987 (DAN-MONICA II) and the second in 1991-1992 (DAN-MONICA III). The MONICA dataset contains a random sample of 4,000 participants and 400 variables from the first three study waves describing questionnaire answers with participant information on socio-demographics, medication, lifestyle and psychological factors, as well as anthropometrics, genetic markers and laboratory values. The binary target variable is `heart_blood_disease` and encodes whether at least one of the following conditions is present in the second follow-up: heart attack, stroke, blood clot in brain, hypertension, other heart disease. Out of 2648 participants, 688 are positive w.r.t. `heart_blood_disease` ($\approx 26\%$).

The characteristics of the datasets are shown in Table 4.2 and Table 4.3.

	Study wave: SHIP-		
	0	1	2
#Participants	886	886	886
Sex [% female]	51.9	51.7	53.2
Age [years]	44.1±12.9	49.9±12.7	56.1±12.6
#Variables	271	61	71
#Shared variables	57		
Target variable	`low`: 58.6%, `intermediate`: 19.8%, `high`: 21.7%		

Table 4.2: Characteristics of the SHIP datasets.

	Study wave: Dan-MONICA-		
	1	2	3
#Participants	2546	2429	2648
Sex [% female]	49.5	-	49.8
Age [years]	44.5±10.8	49.6±10.8	55.4±10.8
#Variables	115	110	197
#Shared variables	69		
Target variable	`negative`: 74.0%, `positive`: 26.0%		

Table 4.3: Characteristics of the MONICA datasets.

4.5.2 Evaluation Setting and Variants

In our experiments we evaluate to what extend the derived Evolution Features from EVO-Extract are able to improve the performance of classification algorithms. Given a classification algorithm, we investigate the following variants:

- We vary the feature sets, providing a variant ”orig” (only the original features), ”evo” (only the Evolution Features), ”smp” (a competing method providing features containing simple statistics w.r.t. the evolution, i.e. mean, variance, mode, median, minimum, maximum), ”orig+evo” (the original and Evolution Features), ”orig+smp” (the original and simple statistics features) or ”all” (all original and derived features).
- We vary the use of a feature selection algorithm (CFS [12]) before the classification step, denoted as ”Feat.Red.”.

As classifiers we consider the following algorithms: Random Forest (RF) [97], Gradient Boosted Trees (GBT) [98], Support Vector Machine (SVM) Linear (SVM-L) [99], SVM Polynomial (SVM-P) [99], C4.5 [100], C5.0 [101], CART [102], Multilayer Perceptron (MLP) [103], Naive Bayes (NB) [104] and One Nearest Neighbor (1NN) [105]. When parameters are expected, we use standard settings. We execute a five fold cross-validation and evaluate on Accuracy, Cohen’s Kappa κ, Sensitivity and Specificity. To compare our 100 variants ($10 * 2 * 5$, sole ”smp” variants excluded) we further calculate a ranking based on the mean of all performance measures. All in all, we execute 400 5-fold cross-validation runs, which equals 2000 evaluation runs in total.

4.5.3 Experiments

Table 4.4 shows the top-25 classification runs and their performance on the female partition of the SHIP dataset ordered on the average achieved rank. The best performing variant uses GBT with CFS on all features (which includes Evolution Features) with an average rank of 22.88, an accuracy of 0.8 and a κ of 0.34. The best algorithms per evaluation measure are Random

Classifier	Feat. Set	Feat. Red.	Acc.	κ	Sens. class 1	Sens. class 2	Sens. class 3	Spec. class 1	Spec. class 2	Spec. class 3	Avg. Rank
GBT	all	CFS	0.80	0.34	0.94	0.20	0.71	0.65	0.95	0.94	22.88
RF	all	CFS	**0.81**	**0.37**	0.96	0.06	**0.82**	0.60	0.99	0.93	22.94
C5.0	orig+smp	CFS	0.79	0.32	0.93	0.15	0.74	0.63	0.95	0.94	24.19
RF	orig+smp	CFS	0.80	0.35	0.96	0.06	0.80	0.58	0.98	0.93	25.62
NB	all	CFS	**0.81**	0.36	0.97	0.05	0.78	0.55	0.98	0.95	26.81
RF	evo	CFS	0.80	0.32	0.95	0.09	0.74	0.57	0.97	0.94	28.31
GBT	orig+smp	CFS	0.79	0.32	0.92	**0.28**	0.70	**0.67**	0.94	0.94	28.56
C5.0	orig+evo	none	0.79	0.31	0.93	0.12	0.78	0.62	0.96	0.92	29.56
SVM-L	evo	CFS	0.79	0.30	0.93	0.09	0.77	0.58	0.97	0.93	31.88
RF	orig+evo	none	0.79	0.31	0.94	0.06	0.77	0.60	0.97	0.92	32.75
NB	evo	CFS	0.79	0.30	0.93	0.15	0.72	0.64	0.96	0.91	32.75
NB	orig+smp	CFS	0.80	0.32	0.96	0.03	0.74	0.50	0.98	0.95	33.00
SVM-P	orig+evo	CFS	0.79	0.32	0.95	0.02	0.81	0.59	0.99	0.91	33.50
NB	orig+evo	CFS	0.79	0.32	0.97	0.02	0.71	0.47	0.99	0.95	34.31
GBT	evo	CFS	0.78	0.29	0.92	**0.28**	0.66	0.65	0.93	0.94	35.19
GBT	orig	none	0.78	0.27	0.92	0.15	0.73	0.62	0.93	0.94	35.69
RF	all	none	0.79	0.30	0.94	0.08	0.75	0.57	0.98	0.92	36.06
RF	orig+evo	CFS	0.79	0.30	0.94	0.03	0.78	0.57	0.99	0.91	36.81
SVM-P	evo	CFS	0.79	0.30	0.96	0.02	0.73	0.53	0.99	0.93	37.56
RF	orig	none	0.79	0.29	0.96	0.03	0.69	0.51	0.98	0.94	39.12
NB	orig	CFS	0.78	0.28	**0.98**	0.02	0.60	0.40	**1.00**	**0.96**	40.12
MLP	all	CFS	0.78	0.29	0.93	0.03	**0.82**	0.59	0.99	0.90	40.44
C4.5	orig+smp	CFS	0.78	0.28	0.92	0.09	0.78	0.56	0.98	0.92	40.81
C5.0	evo	CFS	0.77	0.25	0.90	0.17	0.75	0.60	0.94	0.93	41.31
MLP	orig+evo	CFS	0.78	0.28	0.93	0.00	**0.82**	0.58	**1.00**	0.89	41.38

Table 4.4: Top-25 classification variants for female partition (SHIP), 5-fold cross validation, best values in boldface.

Classifier	Feat. Set	Feat. Red.	Acc.	κ	Sens. class 1	Sens. class 2	Sens. class 3	Spec. class 1	Spec. class 2	Spec. class 3	Avg. Rank
SVM-L	all	CFS	**0.66**	**0.36**	0.87	0.32	0.62	0.70	0.88	0.87	21.81
SVM-P	orig+smp	CFS	0.65	**0.36**	0.89	0.26	0.62	0.69	0.89	0.87	22.00
SVM-L	orig+smp	CFS	0.65	0.35	0.88	0.28	0.60	0.71	0.86	0.87	24.69
RF	orig+evo	CFS	0.65	0.34	0.88	0.22	0.65	0.68	0.89	0.86	27.50
RF	all	CFS	0.65	0.34	0.89	0.21	0.64	0.70	0.89	0.85	28.06
NB	orig+evo	CFS	0.65	0.35	0.92	0.13	0.69	0.62	0.94	0.85	28.38
SVM-P	orig+evo	CFS	0.65	0.34	0.89	0.26	0.60	0.67	0.88	**0.88**	28.44
C5.0	orig+evo	none	0.64	0.33	0.83	0.28	0.67	0.68	0.88	0.87	28.50
SVM-L	orig	CFS	0.65	0.34	0.87	0.26	0.62	0.68	0.88	0.86	28.50
NB	all	CFS	0.65	0.34	0.92	0.10	0.70	0.62	**0.95**	0.85	29.50
SVM-L	orig+evo	CFS	0.64	0.32	0.84	0.29	0.62	0.71	0.86	0.86	30.75
MLP	orig+evo	CFS	0.65	0.34	0.87	0.15	**0.73**	0.72	0.91	0.81	30.75
SVM-P	all	CFS	0.64	0.33	0.89	0.25	0.59	0.67	0.89	0.87	31.44
RF	orig+evo	none	0.64	0.33	0.89	0.14	0.69	0.63	0.93	0.85	32.44
GBT	orig+evo	CFS	0.63	0.31	0.79	**0.34**	0.64	**0.74**	0.84	0.85	33.75
MLP	all	CFS	0.64	0.33	0.88	0.13	0.72	0.69	0.92	0.81	34.50
NB	orig	CFS	0.64	0.33	**0.93**	0.08	0.69	0.64	**0.95**	0.82	34.56
SVM-L	evo	CFS	0.64	0.32	0.88	0.25	0.58	0.66	0.90	0.85	34.81
RF	orig+smp	CFS	0.64	0.32	0.88	0.20	0.64	0.69	0.89	0.84	35.06
SVM-P	orig	CFS	0.64	0.32	0.89	0.25	0.57	0.67	0.88	0.87	35.31
MLP	orig+smp	CFS	0.64	0.32	0.87	0.11	**0.73**	0.71	0.91	0.80	36.00
RF	orig	CFS	0.64	0.32	0.88	0.16	0.66	0.68	0.91	0.83	36.69
RF	evo	CFS	0.63	0.32	0.88	0.17	0.65	0.72	0.89	0.81	36.69
GBT	all	none	0.63	0.31	0.84	0.22	0.67	0.69	0.88	0.84	36.69
GBT	orig	none	0.63	0.32	0.86	0.23	0.63	0.71	0.88	0.83	36.88

Table 4.5: Top-25 classification Variants for male partition (SHIP), 5-fold cross validation, best values in boldface.

Classifier	Feat. Set	Feat. Red.	Acc.	κ	Sens.	Spec.	Avg. Rank
RF	orig+evo	CFS	0.96	0.51	**0.38**	**0.85**	11.88
SVM-L	orig+evo	CFS	0.95	0.53	**0.38**	**0.85**	13.62
SVM-L	all	CFS	0.95	0.52	0.37	**0.85**	14.12
RF	all	CFS	0.96	0.51	0.37	**0.85**	14.38
SVM-P	orig+evo	CFS	0.95	0.51	0.36	**0.85**	16.62
MLP	orig+evo	CFS	0.94	0.57	0.37	**0.85**	17.00
SVM-P	all	CFS	0.96	0.49	0.35	**0.85**	18.50
MLP	all	CFS	0.94	0.57	0.37	**0.85**	19.12
SVM-P	evo	CFS	0.95	0.50	0.34	0.84	20.62
NB	evo	CFS	0.95	0.48	0.33	0.84	22.12
RF	evo	CFS	0.95	0.49	0.33	0.84	23.25
SVM-L	evo	CFS	0.94	0.52	0.32	0.84	23.50
MLP	evo	CFS	0.93	0.57	0.35	0.84	23.88
RF	all	none	0.96	0.43	0.32	0.84	25.38
C5.0	orig+evo	CFS	0.93	0.54	0.32	0.84	27.00
RF	orig+evo	none	**0.97**	0.41	0.30	0.83	27.12
MLP	orig	CFS	0.95	0.47	0.31	0.84	27.75
C5.0	all	CFS	0.92	**0.59**	0.33	0.84	28.50
GBT	evo	CFS	0.94	0.50	0.30	0.83	28.88
MLP	orig+smp	CFS	0.94	0.47	0.30	0.83	29.50
NB	all	CFS	0.95	0.44	0.29	0.83	31.62
RF	evo	none	0.96	0.40	0.28	0.83	32.62
GBT	all	CFS	0.93	0.52	0.29	0.83	34.00
GBT	orig+evo	CFS	0.92	0.55	0.29	0.83	34.75
RF	orig+smp	CFS	0.96	0.42	0.28	0.83	35.00

Table 4.6: Top-25 classification variants for female partition (MONICA), 5-fold cross validation, best values in boldface.

Classifier	Feat. Set	Feat. Red.	Acc.	κ	Sens.	Spec.	Avg. Rank
MLP	orig+evo	CFS	**0.83**	**0.36**	0.51	0.95	13.25
MLP	orig+smp	CFS	**0.83**	0.35	0.51	0.94	14.50
SVM-L	orig+smp	CFS	**0.83**	**0.36**	0.47	0.96	14.75
MLP	orig	CFS	**0.83**	0.35	0.51	0.94	15.25
RF	all	CFS	**0.83**	0.35	0.50	0.95	16.75
SVM-L	all	CFS	**0.83**	0.35	0.46	0.96	18.50
MLP	all	CFS	0.82	0.33	0.51	0.94	20.38
SVM-P	all	CFS	**0.83**	0.35	0.43	**0.97**	22.38
SVM-L	orig	CFS	0.82	0.33	0.44	0.96	22.50
RF	orig+evo	CFS	0.82	0.33	0.47	0.95	22.75
GBT	all	none	0.82	0.33	0.50	0.94	23.00
SVM-L	orig+evo	CFS	0.82	0.33	0.45	0.96	23.00
SVM-P	orig+evo	CFS	0.82	0.34	0.42	**0.97**	24.12
C5.0	all	none	0.82	0.33	0.45	0.95	26.50
SVM-P	orig+smp	CFS	0.82	0.33	0.42	0.96	27.50
GBT	orig+evo	none	0.82	0.31	0.48	0.94	27.75
C5.0	orig+smp	CFS	0.82	0.31	0.52	0.92	28.75
GBT	orig+evo	CFS	0.82	0.31	**0.53**	0.92	29.62
RF	orig+evo	none	0.82	0.32	0.42	0.96	29.75
SVM-P	orig	CFS	0.82	0.32	0.42	0.96	30.25
RF	orig	none	0.82	0.31	0.44	0.95	30.38
C5.0	orig+evo	CFS	0.82	0.31	0.49	0.93	30.50
RF	evo	CFS	0.81	0.31	0.44	0.95	32.62
SVM-P	orig+evo	none	0.81	0.31	0.43	0.96	33.12
RF	evo	none	0.81	0.31	0.43	0.95	33.50

Table 4.7: Top-25 classification variants for male partition (MONICA), 5-fold cross validation, best values in boldface.

Forest (all features, CFS) for Accuracy and κ, Naive Bayes (original features) for class 1 Sensitivity, GBT (original+simple or evo, CFS) for class 2 Sensitivity, Random Forest (all) or MLP (original+evo or all features, CFS) for class 3 Sensitivity, GBT (orig+smp, CFS) for class 1 Specificity, Naive Bayes (original) and MLP (original+evo) for class 2 Specificity and Naive Bayes (original feature set) for class 3 Specificity. The respective winners for three out of the eight evaluation measures use Evolution Features. Although, the other four measures have winners that do not use Evolution Features, all of them exhibit huge performance penalties w.r.t. other evaluation measures, for example Naive Bayes (original, CFS) with 0.02 in class 2 Sensitivity and low class 1 Specificity.

When comparing algorithm variants that use the Evolution Features versus the same variants that do not, it can be observed that the Evolution Features are able to improve the Random Forest, GBT, Naive Bayes, SVM Linear, SVM Polynomial and MLP algorithms (6 out of 10 classifiers). Here, the performance of 75% of the classification algorithms in the top-25 was improved when using Evolution Features. Further, most of the top variants utilize CFS (80% of the top 25 variants). Most of the time, the CFS counterpart performs better even for algorithms that already have in-build feature selection (like Random Forest or C5.0), indicating that for complex real-world datasets a dedicated feature selection algorithm still has its place and cannot be simply replaced.

Table 4.5 shows the classifier performance on the male partition of the SHIP dataset. Here, CFS in conjunction with SVM Linear using all features performs best, with an average rank of 21.81, an accuracy of 0.66 and a κ value of 0.36. With respect to each performance measure, the best performing algorithms are SVM Linear (all features, CFS) for Accuracy and κ, Naive Bayes (using only original features) for class 1 Sensitivity, GBT (original+evo, CFS) for class 2 Sensitivity and class 1 Specificity, MLP (original+evo or original+simple, CFS) for class 3 Sensitivity, Naive Bayes (either using original or all features with CFS) for class 2 Specificity and SVM Polynomial (original+evo, CFS) for class 3 Specificity. For five out of the eight measures the respective winning algorithm had to use Evolution Features. With the exception of SVM Polynomial, most algorithms perform better when Evolution Features are considered (SVM Linear, Random Forest, Naive Bayes, C5.0, MLP, GBT). Like for the female partition, the majority (84%) of the algorithms used in the top 25 variants exploit feature selection and perform generally better than their counterparts that do not utilize the same.

The classification performance on the female partition of the MONICA dataset is displayed in Table 4.6. The best performing variant is Random Forest (original+evo feature set,CFS) with an average rank of 11.88. When considering all evaluation measures individually, Random Forest (original+evo) achieves best on Accuracy, C5.0 (all, CFS) achieves best on κ, a number of different variants tie on Specificity and Random Forest (original+evo, CFS) and SVM Linear (original+evo, CFS) achieve best Sensitivity. In all cases, the respective winners had to use Evolution Features, and all algorithms with Evolution Features performed better than their variant counterparts that did not utilize them. In the top 25, CFS variants are dominant with a relative proportion of 88%.

Finally, the performance on the male partition of the MONICA dataset is shown in Table 4.7. MLP (original+evo, CFS) performed best with an average rank of 13.25. With respect to Accuracy the best performance is tied between numerous variants. However, MLP (original+evo, CFS) and SVM Linear (original+simple, CFS) achieved best κ, SVM Polynomial (all features,CFS) achieved best Specificity and GBT (original+evo, CFS) best Sensitivity. For two of the four evaluation measures the respective winning algorithms had to use Evolution Features. With the exception of SVM Linear, all variants with Evolution Features performed better than their non-Evolution Features counterparts (according to rank). Of the top 25 variants, 72% use CFS.

4.5.4 Discussion on Findings

Our experiments show that EVO-Extract is capable of improving the classification performance on evolving objects for a diverse number of state-of-the-art classifiers. For all datasets, the majority of the best performing variants had to utilize Evolution Features to achieve the respective results. This clearly shows that the evolution of objects can contain vital information for real-world classification problems, which is not sufficiently encoded by simple statistics like mean, variance etc.

Also, experimental results show that the Evolution Features often improve classification of the most difficult class (intermediate class / class 2 for the SHIP datasets and positive class for the MONICA dataset): In the SHIP datasets, the top-3 algorithms w.r.t. class 2 Sensitivity used Evolution Features, with the only exception being GBT (orig+smp) that is tied for the first place on the female partition. On the MONICA dataset the top-16 (female partition) and top-1 (male partition) algorithms w.r.t. Sensitivity use Evolution Features. This again indicates that knowledge is codified that was previously only implicit to the object evolution. However, the performance gain from the Evolution Features for each algorithm depends on the dataset, i.e. if the evolution of objects actually contains relevant knowledge. Keeping this in mind, the average decrease in rank of a top-25 classifier when using Evolution Features is ≈ 3.7 on the SHIP female partition (with SVML-L benefiting the most with an decrease in 9.5 ranks), ≈ 3.8 on the SHIP male partition (with C5.0 benefiting the most with an decrease in 8.33 ranks), ≈ 14.4 on the MONICA female partition (with RF benefiting the most with an decrease in 23.12 ranks) and ≈ 4.8 on the MONICA male partition (with RF benefiting the most with an decrease in 13.63 ranks). Note that in order to eliminate the impact of generally bad runs on this calculation, the worst run (w.r.t. rank) in the top-25 was used as a baseline. Therefore, this performance gain estimate is highly conservative (since the actual decrease in ranks is limited) because Evolution Feature variants dominate the top-25 variants (out of 100 variants) and their non-Evolution Feature counterparts are often absent from the top-25.

A benefit of the derived Evolution Features is the fact that they do not exclude the utilization of other methods that derive features from the evolution of objects. In fact, our experiments show that for some datasets (SHIP datasets) an extensive feature space consisting of all possible derived features (Evolution Features, original features and simple statistics) performed best.

Regarding the specific Evolution Features that have the most impact, we observed that this is actually highly dependent on the respective dataset. Only few similarities in Evolution Feature importance existed across the datasets. In most cases a small but highly important subset of Evolution Features was observed (that was different for all datasets). This supports our approach of deriving a diverse set of Evolution Features with EVO-Extract.

Further, adding Evolution Features to a dataset of evolving objects is highly convenient for a domain expert. No parameter-tuning is required. All parameters are imputed automatically by heuristics that take the data structure into account.

4.6 Conclusion on Methods

We investigated the problem of deriving features from the evolution of objects to codify relevant knowledge for classification tasks. Tackling this problem is important since the evolution of objects can have relevant information w.r.t. a target concept that may improve classification performance and deepen domain understanding if extracted. To do this, we introduced the concept of Evolution Features that are based on (1) sets of sequences which are used to automatically derive relevant, arbitrarily shaped patterns in single time-series features and (2) the change of instances in a multivariate feature-space over time in relation to their previous positions and cluster peers.

Experiments on real-world datasets show that the use of derived Evolution Features improves the classification performance of a majority of tested classifiers. In most cases and for a majority of evaluation measures, classifiers that exploit datasets enriched with Evolution Features outperform counterparts that only employ the original datasets, or datasets that are enriched with conventional evolution-based statistics. This implies that our methods derive features that capture novel target concept associations.

EVO-Extract contains our methods for the codification of relevant knowledge w.r.t. the evolution of objects. By proposing the sub-component to derive pattern-based Evolution Features we answer RQ 1.1. By proposing the sub-components on pre-processing and creation of object- and group-based Evolution Features we answer RQ 1.2. Our experimental results show that enriching datasets with Evolution Features improves classification performance. Therefore, we answer RQ 1 positively.

5. STATIC-Select: Identifying Relevant Features

The second main component of our framework is STATIC-Select. STATIC-Select deals with the task of identifying relevant features from the feature space that is used to explicitly describe an object as motivated in Section 1.1.1. In this chapter we address the following research question:

RQ 2: How to identify the relevant features of evolving objects by exploiting background knowledge?

STATIC-Selects expects a dataset D of evolving or non-evolving objects to be applied onto. For the given input dataset, STATIC-Select conducts the pre-processing step as described in Section 4.1, which leads to an object dataset represented by two (in the case of an evolving object dataset) or one static and normalized feature sets. Since the pre-processing component will transform an evolving object dataset into the same data scheme of a static object dataset, we will only refer to a dataset D with static feature set F as a representative of any output of the pre-processing component. Additionally, STATIC-Select either requires ground-truth in the form of a training dataset with known class-label or a number of ML and NL similarity constraints. In the case of class-label data, STATIC-Select can execute traditional feature selection algorithms to identify subsets of relevant features. If no class-label data, or only few labels exist, STATIC-Select is able to alternatively utilize few instance-level constraints to achieve the same goal.

In either case, the output of STATIC-Select is a subset of relevant features w.r.t. the target concept.

This chapter begins with the problem definition of STATIC-Select. In Section 5.2 we briefly show how relevant subsets of features are identified when traditional labeled training data is available. In the case that this kind of knowledge is not present or desired, we propose our constraint-based subspace clustering algorithm DRESS in Section 5.3 for relevant feature subset identification as well as a method for the validation of the results of DRESS. Then, Section 5.4 shows the evaluation of the component with the help of classification methods and validates the findings of DRESS. At last, Section 5.5 concludes this chapter and provides answers to the research question. Parts of Section 5.2 - Section 5.5 are from our previous publications [27, 30–32] (with modifications).

5.1 Problem Definition

Let $model(D_{tr}, F)$ be a classification model learnt on training data $D_{tr} \subseteq D$ and feature set F, and $perf(model, D_{te})$ be the classification model performance that is obtained when using classification model $model$ to classify each instance $x \in D_{te} \subseteq D$ (such as Accuracy, Specificity etc., we present the utilized evaluation measures as well as an aggregated ”rank” of these measures that may be used as a decision basis in Section 5.4.2). Let further denote $\mathcal{CONSTS}$ as the set of given instance-level similarity constraints and F^* a non-empty subset of relevant features (regarding the target concept) of F.

The goal of STATIC-Select is to use $\mathcal{CONSTS}, F, D$ to find a subset of relevant features F^* for D and the concept described by $\mathcal{CONSTS}$, i.e. $STATIC - Select(\mathcal{CONSTS}, F, D) = F^*$, so that

$$perf(model(D_{tr}, F^*), D_{te}) > perf(model(D_{tr}, F), D_{te}) \tag{5.1}$$

holds true.

Note that this problem definition only refers to our proposed constraint-based subspace selector in STATIC-Select. We omit a problem definition for the implemented CFS-method since it utilizes traditional class-labeled training data and is well studied in literature.

5.2 Correlation-based Feature Selection

STATIC-Select differentiates between two cases when identifying sets of relevant features. When a sufficient number of labeled training data is available (i.e. at least 50% of instances are labeled), STATIC-Select executes a traditional feature selection algorithm. In this implementation of the framework, we opt for the well-studied and reliable Correlation-based Feature Selection (CFS) [12] algorithm. CFS is based on the notion of "Merit" for a feature set F, defined as $M(F)$:

$$M(F) = \frac{|F|\overline{r_{class,F}}}{\sqrt{|F| + |F|(|F|-1)\overline{r_{F,F}}}} \ , \tag{5.2}$$

where $\overline{r_{class,F}}$ denotes the mean feature-class dependence and $\overline{r_{F,F}}$ the mean feature-feature dependence between all features in F. For a given dataset D, these dependencies are measured with the help of the Symmetrical Uncertainty Coefficient (SUC), using the normalized Information Gain (IG). Assume a nominal feature f with the corresponding set of distinct values $VAL(f) = \{val_1(f), ..., val_n(f)\}$, along with a function $split(f, D) = \{D_1, ..., D_n\}$, where $D_i = \{x \in D | f(x) = val_i(f)\}$, that yields the resulting partitions (sets of instances) when splitting the instances in D on the nominal values of f. Then, the SUC between the nominal features f_1 and f_2 in D is computed as follows:

$$SUC(f_1, f_2, D_{tr}) = 2 \times \frac{IG(VAL(f_1), split(f_2, D_{tr}))}{H(VAL(f_1), D_{tr}) + H(VAL(f_2), D_{tr})}, \tag{5.3}$$

where $IG()$ and $H()$ are calculated as shown in Section 4.3.2 on the set of training data D_{tr}.

The CFS-algorithm measures the Merit for different feature sets by conducting a forward-selection, beginning with an empty feature set. In each iteration, the feature that leads to the new highest Merit value for this set is added. The algorithm stops and returns the current feature set when a local optimum for Merit is found, i.e. as soon as the addition of a new feature would not increase the Merit. Algorithm 3 shows this procedure.

```
**Data:** Training dataset $D_{tr} \subseteq D$, set of static features $F$
**Result:** Relevant feature subset $F^*$
$F^* \leftarrow \emptyset$ ; // Initialize empty set of relevant features
$F^{it} \leftarrow F$ ; // Initialize copy of complete feature set
$M_{opt} \leftarrow 0$ ; // Initialize best Merit-value
repeat
    $M_{old} \leftarrow M_{opt}$;
    for each $f \in F^{it}$ do
        $M_{new} \leftarrow Merit(F^* \cup \{f\})$; // Calculate Merit when
          adding $f$ to $F^*$
        if $M_{new} > M_{opt}$ then
            $M_{opt} \leftarrow M_{new}$; // Update $M_{opt}$
            $f^{it} \leftarrow f$ ; // Store the feature that led to a
              better Merit
        end
    end
    if $M_{opt} \neq M_{old}$ then
        $F^* \leftarrow F^* \cup \{f^{it}\}$; // Add $f^{it}$ to $F^*$
        $F^{it} \leftarrow F^{it} \setminus \{f^{it}\}$ ; // Remove $f^{it}$ from $F^{it}$, that was
          added to $F^*$ in the last iteration
    end
until $M_{opt} == M_{old}$;
```

Algorithm 3: CFS with forward-selection.

Note that continuous features must be discretized beforehand, otherwise the Merit cannot be calculated.

5.3 Constraint-based Feature Subspace Selection

In the case that no class-label data is available (or only a limited amount), STATIC-Select executes the Constraint-based Subspace Clustering algorithm DRESS (**D**eriving **R**elevant **E**xample-constrained **S**ub**S**paces) to derive relevant feature subspaces. As input, DRESS requires the instance dataset D, feature set F from which a relevant subspace should be derived from, as well as a set of ML and NL constraints between pairs of instances. The goal of DRESS is then as follows:

> Given dataset D with static feature set F and a set of ML/NL constraints, find subspaces $F^* \subseteq F$ which best describe the concept, as reflected in the constraints, where "best" refers to similarity/separation of instances under ML and NL constraints as well as constraint satisfaction.

Constraints as described in Section 2.1.2 should reflect the similarity (ML constraints) or dissimilarity (NL constraints) of instances w.r.t. the target concept (i.e. the known or unknown class variable). DRESS assumes that we can find groups of similar instances in the data (natural clusters), in specific feature subspaces, that satisfy these constraints. If the given constraints reflect the target concept well, than the found subspaces that contain these groups should be relevant w.r.t. the target concept. DRESS translates this assumption by searching for subspaces where ML-constrained instances have high pairwise similarity to each other and NL-constrained instances exhibit high pairwise dissimilarity.

The general workflow of DRESS is shown in Figure 5.1. Constraints can be given by a domain expert, or they can be automatically derived from class-labeled data. Using the constraints, DRESS iteratively merges subspaces (starting with subspaces of cardinality one) to find new, unvisited subspaces that contain relevant clusters (clusters that satisfy constraints), i.e. finding subspaces that score better than previously visited subspaces according to its underlying quality function. At the beginning of each iteration the currently "best" subspace F^{best} in the candidate set is chosen and then removed from the set of candidate subspaces. Then, DRESS extends F^{best} with each remaining candidate individually to create the set of potential new candidate subspaces (merging step). To reduce complexity, the intermediate filtering step drops subspaces (potential candidates) that are unlikely to exceed the current quality threshold. Afterwards, the algorithm computes the complete quality scores of the remaining potential candidate subspaces with the help of a clustering method. If the quality of a subspace exceeds the highest yet observed quality q_{best}, DRESS retains it as a candidate subspace for further extension, updates q_{best} and stores all contained clusters. Each time a successful subspace merge yields a new candidate subspace, the subspaces used for merging are removed from the candidate set. DRESS terminates as soon as the candidate subspace set is empty, i.e. no "better" subspace was found, and returns the set of all stored subspaces with their associated clusters. The quality computation and clustering are described in Section 5.3.2 and Section 5.3.3. "Merging" and "Filtering" are explained in Section 5.3.4.

5.3.1 Constraint Generation

The constraints used by DRESS can be generated in different ways. Constraints should reflect similarity w.r.t. the target concept between instances. One of the simplest methods to generate such constraints is to use instances for which the class is known (or, if unknown, to ask a domain expert to label a random number of instances, for example as "healthy" and "ill" if the target concept is a disease) and then automatically generate a ML constraint between each pair of instances with the same label, and a NL constraint between each pair of instances with different label.

As an alternative one can adopt the concept presented by Amid et al. [106]: Assume that a number of labeled instances already exists. A domain expert is presented with two instances, x_1 and x_2, that are dissimilar w.r.t. the target concept, and a third instance x_3. Then, the expert is asked to assign x_3 to either x_1 or x_2, the one that is in his opinion more similar to x_3. This can be repeated multiple times and from each expert assignment a ML constraint may be derived, as well as a NL constraint from the preselected instances.

Basically, there is no strict protocol to follow in order to derive a number of constraints. All that is necessary is the decision of the expert on what

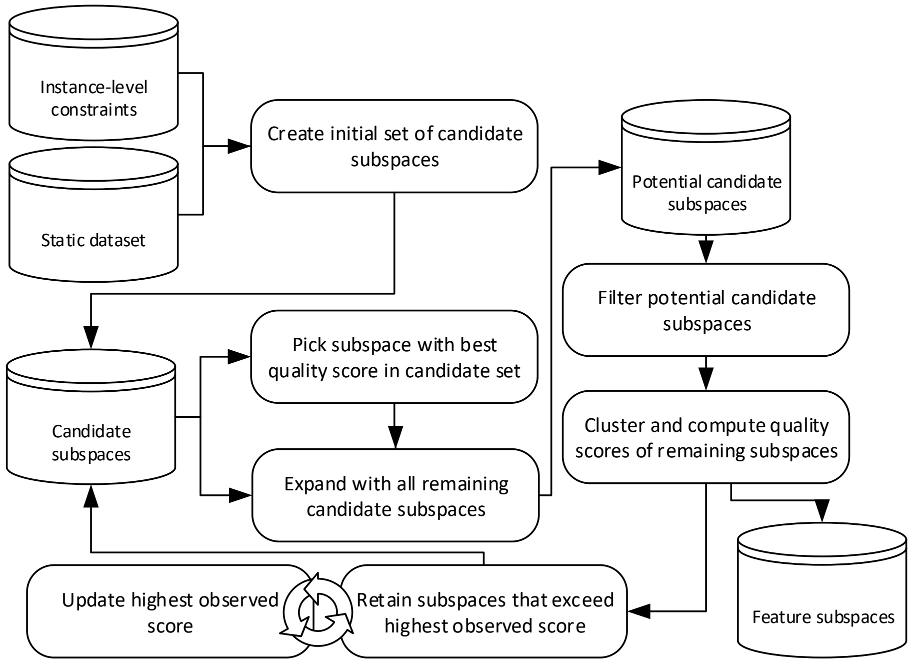

Instance-level constraints
Static dataset
Create initial set of candidate subspaces
Potential candidate subspaces
Filter potential candidate subspaces
Candidate subspaces
Pick subspace with best quality score in candidate set
Expand with all remaining candidate subspaces
Cluster and compute quality scores of remaining subspaces
Feature subspaces
Update highest observed score
Retain subspaces that exceed highest observed score

Figure 5.1: Overview of the DRESS workflow.

he deems to be similar w.r.t. target concept and the instances, or a small portion of ground-truth.

5.3.2 Subspace Quality

DRESS evaluates subspaces in a semi-supervised way as it investigates the structure of the complete dataset in feature subspaces. For this, DRESS utilizes the provided ML and NL instance-level constraints , i.e. the background information w.r.t. the target concept, from a small number of instances. The quality function used to score (and rank) candidate subspaces in DRESS is based on the clustering results within the respective subspace and takes the following criteria into account: (1) whether the data exhibits a satisfactory structure within a space w.r.t. the given constraints and (2) the proximity between constrained instances in the subspace. Criterion (1) is necessary to drop spaces without satisfactory clusters. DRESS assumes that if a feature subspace is relevant w.r.t. the concept, one or more corresponding natural structures in such space can be found that satisfy the given constraints. These clusters should be distinct and not diluted, which means that reasonable parameter settings of the clustering algorithm should lead to their detection, i.e. no parameter optimization should be necessary. Because the constraints reflect background knowledge regarding the similarity of the instances w.r.t. the target concept, good subspaces should contain clusters with objects under ML-constraints, and separate objects under NL-constrains (putting them in different clusters). For (1), DRESS calculates the constraints satisfaction of a clustering as follows: Given subspace F^{sub}, let $\mathcal{ML}_{sat}(F^{sub})$ be the set of satisfied ML constraints and $\mathcal{NL}_{sat}(F^{sub})$ be the respective set for the NL constraints. Then

$$q_{cons}(F^{sub}) = \frac{|\mathcal{ML}_{sat}(F^{sub})| + |\mathcal{NL}_{sat}(F^{sub})|}{|\mathcal{ML}| + |\mathcal{NL}|} \tag{5.4}$$

defines the constraint satisfaction within F^{sub}, where higher values indicate higher quality.

When exclusively relying on q_{cons}, DRESS faces the problem of ignoring the continuous similarity between the constrained instances. Imagine two spherical clusters in two different subspaces that both satisfy the same Must-Link constraint ML_1, where in one subspace the ML_1 constrained instances have almost identical feature values and in the other subspace these instances lie on opposite border regions of the same cluster. In this scenario, q_{cons} for both subspaces is identical. But in reality the first subspace should be scored better than the latter. This problem is avoided by accounting for criterion (2) and defining a q_{dist}, that incorporates the similarity between instances under constraints in a subspace through distance calculations. We define q_{dist} as the difference in the average distances between NL and ML pairs of instances in the respective subspace: Let $d(\pi_{F^{sub}}(x_1), \pi_{F^{sub}}(x_2))$ be the distance between instance x_1 and x_2 in F^{sub}. DRESS computes q_{dist} of F^{sub} as

$$q_{dist}(F^{sub}) = d_{avg}(F^{sub}, \mathcal{NL})/(d_{avg}(F^{sub}, \mathcal{ML}) + d_{avg}(F^{sub}, \mathcal{NL})), \quad (5.5)$$

$$\text{with } d_{avg}(F^{sub}, \mathcal{NL}) = \sum_{\{x_1,x_2\} \in \mathcal{NL}} d(\pi_{F^{sub}}(x_1), \pi_{F^{sub}}(x_2))/|\mathcal{NL}|,$$

$$\text{and } d_{avg}(F^{sub}, \mathcal{ML}) = \sum_{\{x_1,x_2\} \in \mathcal{ML}} d(\pi_{F^{sub}}(x_1), \pi_{F^{sub}}(x_2))/|\mathcal{ML}|,$$

where higher q_{dist} values indicate higher quality (note that the distance function must be symmetric). Equation 5.5 promotes subspaces where ML constrained instances are closer (i.e. more similar) to each other in comparison to NL constrained instances. However, to compute q_{dist} we must deal with the possible heterogeneity of a given dataset. This is accomplished by utilizing the Heterogeneous Euclidean Overlap Metric (HEOM) as distance function (Equation 4.5 in Section 4.3). The final quality function of DRESS for subspace scoring is then given as

$$q(F^{sub}) = q_{cons}(F^{sub}) \cdot q_{dist}(F^{sub}). \quad (5.6)$$

5.3.3 Subspace Clustering

DRESS uses DBSCAN (like EVO-Extract in Chapter 4) to find clusters in different subspaces. For each (potential) candidate subspace F^{cand}, DRESS invokes DBSCAN once on $\pi_{F^{cand}}(D)$. To perform clustering, DRESS automatically calculates a set of reasonable parameter settings according to the knee-point method as described in Section 4.3.1 (again with $MinPts = round(ln(|D|))$, as suggested by Ester et al. [48]). Contrary to our Evolution

Feature method, DRESS does not use sequence-examples. Instead, DRESS calculates the *Eps*-parameter value in a feature subspace F^{sub} by applying the knee-point method on the dataset projection in $\pi_{F^{sub}}(D)$. This corresponds to substituting a sequence-example of an time-series feature with an instance projection in a feature subspace.

5.3.4 Merging and Filtering of Subspaces

DRESS initializes the set of candidate subspaces with all spaces of cardinality one as depicted in Algorithm 4: Let F be the set of all static features in dataset D, the initial candidate set is defined as

$$\mathcal{F}^{cand} = \left\{ F^{cand} | F^{cand} \subseteq F \wedge |F^{cand}| = 1 \right\}.$$

During initialization, the quality $q(F^{cand})$ of each subspace $F^{cand} \in \mathcal{F}^{cand}$ is calculated, stored and all resultant clusterings are saved. For the general subspace candidate generation and filtering process see Algorithm 5: Here, q_{best} is initialized as the highest observed quality among the initial set of candidate subspaces of cardinality one (which is given by Algorithm 4). Then, DRESS iteratively chooses the subspace $F' \in \mathcal{F}^{cand}$ that has the highest quality $q()$ among the subspaces in $\mathcal{F}^{cand}$ and sets $\mathcal{F}^{cand} = \mathcal{F}^{cand} \backslash F'$ (removing the chosen space from the candidate set). DRESS builds new potential candidate spaces $F^{new} = F' \cup F''$ by merging F' with all remaining subspaces $F'' \in \mathcal{F}^{cand}$ in an effort to find subspaces with a $q()$ that beats q_{best}. To limit the number of subspaces that must be fully scored (calculating both q_{cons} and q_{dist}), DRESS uses a filter condition based on the less complex part q_{dist} of the quality function, that prevents the clustering step of merged subspaces that probably do not contribute to an improvement w.r.t. the full quality $q()$: For each new potential subspace candidate $F^{new} = F' \cup F'$ DRESS picks the subspace $F^{dist} \in \{F'', F'\}$ with higher q_{dist},

$$F^{dist} = \begin{cases} F', & \text{if } q_{dist}(F') \geq q_{dist}(F''), \\ F'', & \text{otherwise,} \end{cases}$$

and performs clustering in F^{new} if, and only if, the following filter condition is satisfied:

$$(\Delta(F^{new}, F^{dist}, \mathcal{NL}) - \Delta(F^{new}, F^{dist}, \mathcal{ML})) > 0,$$

```
Data: Dataset $D$, feature set $F$, $ML$ and $NL$ constraints sets
      $\mathcal{ML}$ and $\mathcal{NL}$
Result: Map of subspaces and contained clusterings $clusterMap$,
        set of candidate subspaces $\mathcal{F}^{cand}$, map of subspace
        quality scores $qualityMap$
$clusterMap \leftarrow newMap < Subspace, Clustering > ();$
 // Initialize empty map where clusterings are stored
 and mapped to the subspaces they were identified in
$qualityMap \leftarrow newMap < Subspace, Double > ();$
 // Initialize empty map where quality scores are
 stored and mapped to the subspaces
for each $f \in F$ do
    $\mathcal{C}_{\{f\},D} \leftarrow \text{DBSCAN}(\pi_{\{f\}}(D));$ // Cluster with knee-point
     method in $\pi_{\{f\}}(D)$
    $clusterMap.put(\{f\}, \mathcal{C}_{\{f\},D});$ // Store initial clusters
     in map
    $\mathcal{F}^{cand} \leftarrow (\mathcal{F}^{cand} \cup \{f\});$
    // store subspace candidate
    $qualityMap.put(\{f\}, calcQuality(\mathcal{C}_{\{f\},D}, \mathcal{ML}, \mathcal{NL}));$
     // Calculate and store the quality of clustering
     $\mathcal{C}_{\{f\},D}$, according to Equation 5.6, into a map that
     references the subspace
end
```

Algorithm 4: DRESS candidate subspace initialization. We denote as $C_{F,D}$ a clustering found in subspace F on D.

$$\text{with } \Delta(F^{new}, F^{dist}, \mathcal{CONS}) = d_{avg}(F^{new}, \mathcal{CONS}) - d_{avg}(F^{dist}, \mathcal{CONS}),$$

This condition checks whether the dissimilarity in the new subspace between NL constrained instances increases more than for ML constrained instances, i.e. if the space leads to a better separation between them. If a new potential candidate subspace F^{new} satisfies this condition, DRESS commences with clustering in it, computing its complete quality $q()$ in the process, and stores all found clusters. After the full quality assessment, if it is found that $F^{new} = F' \cup F''$ satisfies $q(F^{new}) > q_{best}$, DRESS sets $\mathcal{F}^{cand} = \mathcal{F}^{cand} \setminus F''$ to eliminate the lower quality subspace from the candidate set (preventing further merging with it) and adjusts the current best quality value to $q_{best} = q(F^{new})$. Newly discovered high-quality subspaces (that improved q_{best}) are inserted in the candidate set $\mathcal{F}^{cand}$ for further merging, setting $\mathcal{F}^{cand} = \mathcal{F}^{cand} \cup \{F^{new}\}$. When all F^{new} that passed the filter criterion have been scored, a new iteration begins, again selecting the best subspace that is still in the candidate set and repeating all aforementioned steps. DRESS terminates when the candidate set is empty, i.e. no F^{new} with higher quality than the current q_{best} is found. It returns a list of all subspaces which have been fully scored (which includes their $q()$ value) and the clusterings associated with the spaces.

5.3.5 Subspace Cluster Validation

Our proposed framework includes an optional component for the validation of subspace clusters that were found by DRESS in relevant subspaces. DRESS uses these clusters to evaluate subspaces towards their relevance w.r.t. the target concept on completely independent datasets. Given a set of subspace clusters $\mathcal{C}$ found by DRESS and a validation dataset VD that is *comparable* to the set DD used for the discovery of these clusters, the validation component investigates whether the clusters of $\mathcal{C}$ are also present in VD and whether they also exhibit a distinct class distribution in VD. This allows to check if features associated with distinct (regarding the class variable) clusters are generalizable w.r.t. to class relevance in a second dataset, i.e. if relevant features of one subspace in a dataset are also relevant in an independent dataset.

To achieve that, DRESS conducts two steps. First the respective DD and VD are matched with respect to expert-defined covariates to allow fair comparisons between them. The matching accounts for any potential bias of the

```
Data: Dataset $D$, map of subspaces and contained clusterings
      $clusterMap$, set of candidate subspaces $\mathcal{F}^{cand}$, map of
      subspace quality scores $qualityMap$
Result: Map of subspaces and contained clusterings $clusterMap$,
        associated map of subspace quality scores $qualityMap$
$F' \leftarrow pickBest(\mathcal{F}^{cand}, qualityMap)$; // Pick best subspace
 from candidate set according to it's quality value
$q_{best} \leftarrow q(F')$; // Initialize best quality value
$\mathcal{F}^{cand} \leftarrow (\mathcal{F}^{cand} \setminus \{F'\})$; // Clean candidate subspaces set
while $|\mathcal{F}^{cand}| > 0$ do
    for each $F'' \in \mathcal{F}^{cand}$ do
        $F^{new} \leftarrow (F'' \cup F')$; // Merge
        if $q_{dist}(F') > q_{dist}(F'')$ then $F^{dist} \leftarrow F'$;
        else $F^{dist} \leftarrow F''$;
        $q_{dist}(F^{new}) \leftarrow calcDistQual(\pi_{F^{new}}(D))$; // Calculate
         $q_{dist}$ for $F^{new}$
        if $q_{dist}(F^{new}) > q_{dist}(F^{dist})$ then
            // Filter criterion
            $\mathcal{C}_{F^{new},D} \leftarrow DBSCAN(\pi_{F^{new}}(D))$; // If filter
             criterion is passed, cluster $D$ in the
             potential candidate space $F^{new}$ using the
             knee-point method (in this subspace)
             according to Section 4.3.2
            $clusterMap.put(F^{new}, \mathcal{C}_{F^{new},D})$; // Store subspace
             and associated clustering
            $qualityMap.put(F^{new}, calcQuality(\mathcal{C}_{F^{new},D}, \mathcal{ML}, \mathcal{NL}))$;
             // Store $q()$ of subspace
            if $q(F^{new}) > q_{best}$ then
                $q_{best} \leftarrow q(F^{new})$;
                $\mathcal{F}^{cand} \leftarrow (\mathcal{F}^{cand} \cup F^{new})$;
                $\mathcal{F}^{cand} \leftarrow (\mathcal{F}^{cand} \setminus \{F''\})$; // Clean
            end
        end
    end
    $F' \leftarrow pickBest(\mathcal{F}^{cand}, qualityMap)$;
    $\mathcal{F}^{cand} \leftarrow (\mathcal{F}^{cand} \setminus \{F'\})$; // Clean
end
```

Algorithm 5: DRESS subspace processing. We denote as $C_{F,D}$ a clustering found in subspace F on D.

covariates on validation measure scores. To do the match, nearest neighbor propensity score matching [107] is utilized. Here, the original discovery dataset DD and validation dataset VD are combined and a logistic regression model is learned where the dichotomous target variable indicates whether an instance / participant is a member of DD or VD. Given each instance, a propensity score (probability for dataset membership) is calculated by the model. Based on this score, the model matches each DD instance to the VD instance with the most similar score. This procedure yields two datasets (matched DD and VD instances) of the same cardinality: The algorithm unflags all DD and VD instances first. Then one unflagged DD instance x is selected in each iteration. For this instance, the algorithm searches whether an unflagged VD instance x' exists within a similar propensity score threshold. If no instance exist for which this holds true, x is removed from DD. Otherwise, both x and x' are flagged and remain in DD and VD. This concludes a single iteration. As soon as all iterations are finished, i.e. each retained DD instance has been flagged, all remaining unflagged VD instances are removed (”surplus” VD instances that are dissimilar to the remaining DD instances). Output is the set of remaining DD instances (matched discovery dataset) and the set of remaining instances in VD (matched validation dataset). For propensity score matching, the expert is required to specify a maximum propensity score distance threshold. Higher thresholds lead to more matched instances, however the similarity regarding the covariates might be lower than for a lower threshold. Algorithm 6 summarizes the basic propensity score matching procedure.

In the second step, DRESS transfers subspace clusters found in the matched discovery dataset (DD) on the matched validation dataset (VD), to ultimately assess the agreement w.r.t. class distribution and significance. The general workflow of the involved tasks for this second validation step is depicted in Figure 5.2. For each cluster $C \in \mathcal{C}$ of DD, the following procedure is executed individually: First, the neighborhoods are extracted from the core points of C by storing the position of each core point, the *Eps* parameter of the clustering that produced C and the subspace F' where C was found. Then, these neighborhoods are transferred by projecting VD onto F', thus creating $\pi_{F'}(VD)$, and inserting one dummy point p on each position of an original core point in C. Following this, the matched cluster C' is created: For each instance $x \in VD$ the module checks if there exists at least one dummy point p where the distance between x and p is less or equal

```
Data: Discovery dataset $DD$, validation dataset $VD$, set of
      covariates $COV$, propensity margin $margin$
Result: Matched discovery dataset $DD'$, matched validation
        dataset $VD'$
$DD' \leftarrow \emptyset$ ; // Initialize matched $DD$ dataset
$VD' \leftarrow \emptyset$ ; // Initialize matched $VD$ dataset
$mod \leftarrow learnModel(\pi_{COV}(DD \cup VD), target = DD\text{-membership})$
 ; // Learn logistic regression model from instances in
 $DD$ and $VD$ on covariates, with the target variable
 indicating membership in $DD$
$pScoreVD\_Map \leftarrow newMap < Double, Instance > ()$ ;
 // Initialize empty map where each propensity score is
 mapped to the respective instance
for each $x \in VD$ do
    $pScore \leftarrow predict(mod, \pi_{COV}(x))$; // Use learned model to
     calculate propensity score of a $VD$ instance based
     on the covariate-subspace
    $pScoreVD\_Map.put(pScore, x)$; // Store instance and
     score in map
end
for each $x \in DD$ do
    $pScore \leftarrow predict(mod, \pi_{COV}(x))$; // Calculate propensity
     score of $DD$ instance
    $nnScore \leftarrow findNearestScore(pScore, pScoreVD\_Map)$;
     // Find $VD$ instance score most similar to $DD$
     instance score
    if $|nnScore - pScore| < margin$ then
        $DD' \leftarrow DD' \cup \{x\}$; // If a $VD$ instance within
         propensity score margin is found, add $DD$
         instance to $DD'$
        $x' \leftarrow pScoreVD\_Map.get(nnScore)$ ; // Get instance $x'$
         matched to $x$
        $VD' \leftarrow VD' \cup \{x'\}$; // Add matched instance to $VD'$
        $pScoreVD\_Map.remove(pScore, x)$; // Remove map
         entry to avoid duplicate instances in $VD'$
    end
end
```

Algorithm 6: Propensity score matching using a propensity margin with nearest neighbor search.

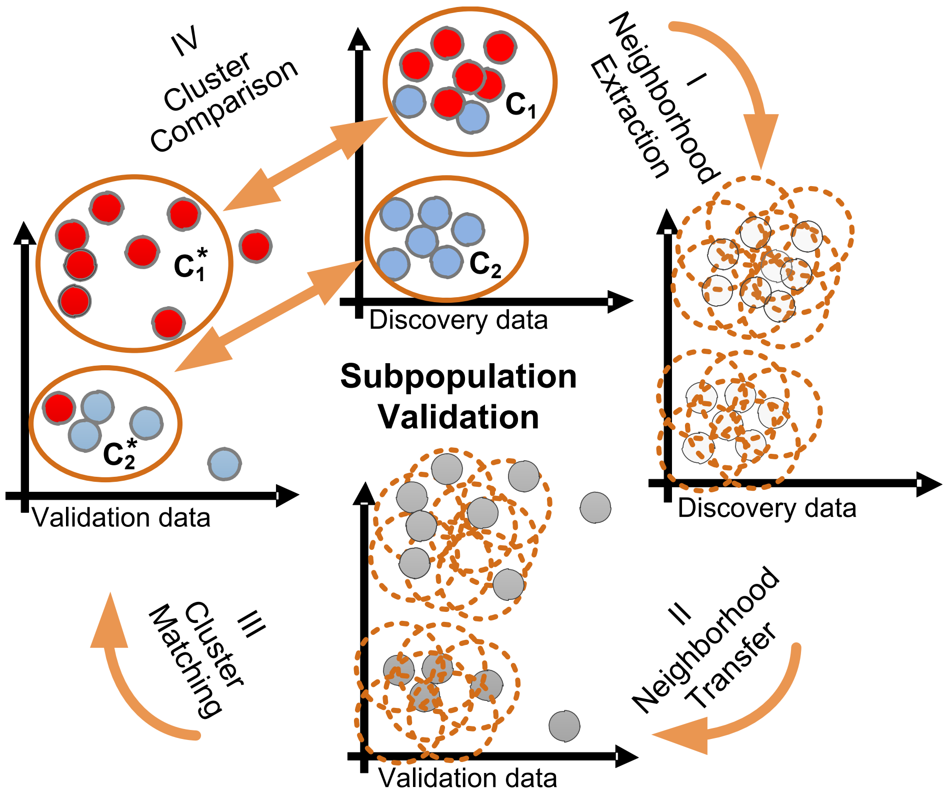


Figure 5.2: Cluster validation scheme. Colors represent class membership.

than the Eps-value that was used to create C, i.e. $d(\pi_{F'}(x), \pi_{F'}(p)) \leq Eps$ in VD. All VD instances that satisfy this criterion form the cluster counterpart C' of C. At last, for each C and C', the class distribution, cluster size and p-value (with H_0 : class distribution of cluster members is equal to non-cluster members) are calculated, stored and compared.

Note that it is possible to use the validation component with any density-based clustering result, i.e. it can be used outside of STATIC-Select (as depicted in Figure 2.3) with results that do not originate from DRESS.

5.4 Evaluation

In this section, STATIC-Select is evaluated specifically in context of our contribution DRESS as follows: Given a classification problem, how do naive classifier (i.e. classifier without in-build feature selection) perform when using the feature subspace that is returned by DRESS in comparison to the exploitation of the complete feature space and the feature space returned by competing feature selection algorithms? To answer this question, we evaluate multiple state-of-the-art feature selection algorithms w.r.t. the performance of different classifiers that utilize their output on two datasets with different (but small) quantities of training data. In the second part we evaluate the validation component by investigating the transferability of DRESS's results, i.e. whether identified relevant cluster can be discovered in independent datasets.

Note that since the conventional feature selection algorithm CFS in STATIC-Select has been thoroughly studied in the past, we will not discuss its performance in the remainder of this section. However, CFS is included as a competing algorithm to DRESS in the results.

5.4.1 Datasets for Feature Subspace Evaluation

For the evaluation of the returned feature subsets of DRESS w.r.t. classification problems we utilize the female and male partitions of SHIP with the outcome ”hepatic steatosis”. Since STATIC-Select works only on datasets consisting of static objects, we only utilize the static features of the datasets as well as the last observations of time-series features (SHIP-2 time-series feature observations) for the feature subspace selection.

5.4.2 Evaluation Setting and Variants on Feature-Subspace Selection

To evaluate the suitability of feature subspaces for a classification task, we adopt the scheme presented by Sheikhpour et al. [108] and compare the performance of DRESS with competing feature selection methods by providing classifiers the resulting feature subspaces and subsequently investigating their classification performance. As classifiers we choose the most basic and naive algorithms kNN [105] and Naive Bayes [104] since we want to exclude algorithms that already perform well on the complete feature space because of in-build feature selection mechanism or regularization (i.e. tree-based algorithms, SVM) which would bias results in favor of feature selection methods that do not select any features at all.

For each classifier, we evaluate its performance on the complete datasets as well as with the following previous feature subspace selection methods:

- $MI1$: Feature-selection based on the features Mutual Information to the class variable. Selects the 50% of features with highest Mutual Information towards the class variable [24].
- RFI: Feature selection based on the Mean Decrease Impurity that is calculated across a tree-ensemble build on the training data. Selects the top 50% of features that contribute to the impurity decrease in the complete ensemble most [109].
- CFS: Correlation-based Feature Selection as described in Section 5.2.
- $CONS$: A semi-supervised feature selection algorithm that scores each feature according to its Laplacian Score and Constrained Laplacian Score. Selects the 50% top-ranked feature [69].
- $DRESS$: Our DRESS algorithm from Section 5.3.
- $Plain$: The base-classifier without a prior feature selection.

Regarding the kNN-classifier, we choose for each variant the $k \in \{5, 7, 9, 10\}$ that performed best on the complete training dataset.

For $CONS$ we use standard parameter settings (bandwidth $t = 10$, and $k = 5$ to assume the five nearest instances of a query instance as neighbors of the query instance to calculate the Laplacian Scores for the unlabeled data) [69].

We derive all possible instance-level constraints for DRESS and CONS from the labels of a small subset of instances in the respective training data fold of a classifier (cf. Section 5.3.1) before learning a classification model. The competing feature selection algorithms are also applied on the same subset of training data before classification model learning. Training-fold subsets that were used for feature selection contain 6,8,10,12,14,16,18 and 20 random training-instances. Afterwards, the resulting feature subspace of DRESS and competing algorithms are used to reduce the feature space in the current training and test fold. We calculate Accuracy, Kappa, Sensitivity and Specificity over a ten-fold cross-validation for each subset of training instances (equaling 80 evaluation runs per variant and dataset) and report on the classifiers results on the datasets. At last, we calculate a rank for each variant and evaluation measure pair that represents the average variant position in the top-6 across all evaluation runs for a dataset. Ideally, our proposed method improves classification performance w.r.t. the most difficult class (the positive class) while preserving good overall Accuracy and Kappa, i.e. it selects features that are informative to both the minority and majority class.

5.4.3 Results on Feature Subspace Selection

Figure 5.3 and Figure 5.4 show the Accuracy, Kappa, Sensitivity and Specificity line- and boxplots for all feature selection variants using the kNN (Figure 5.3) and Naive Bayes (Figure 5.4) classifier on the female partition of the SHIP dataset. Table 5.1 shows the average rank of each variant on the female partition of SHIP across all runs and classifiers. Regarding kNN, $DRESS$ achieves overall highest observed Sensitivity (60.2%) with 18 training instances. Additionally, $DRESS$ outperforms all competitors except $CONS$ for Kappa and outperforms all variants except $CONS$ and $Plain$ on Accuracy. Performance variance of $DRESS$ is among the lowest for all evaluation measures compared to its competitors.

For Naive Bayes, the Accuracy values of RFI, $CONS$ and $DRESS$ are close ($\approx 72.5\%$) on 10,12 and 18 training instances. However, $DRESS$ again achieves overall highest observed Sensitivity (41.6%) with 18 and 20 training instances. In addition, $DRESS$ outperforms competitors on Accuracy for 12 and 18 training instances, on Kappa for all settings except 6, 14 and 20 training instances and on Sensitivity on all training instance subsets. Performance variance of $DRESS$ is comparable to that of CFS, $MI1$ and $CONS$.

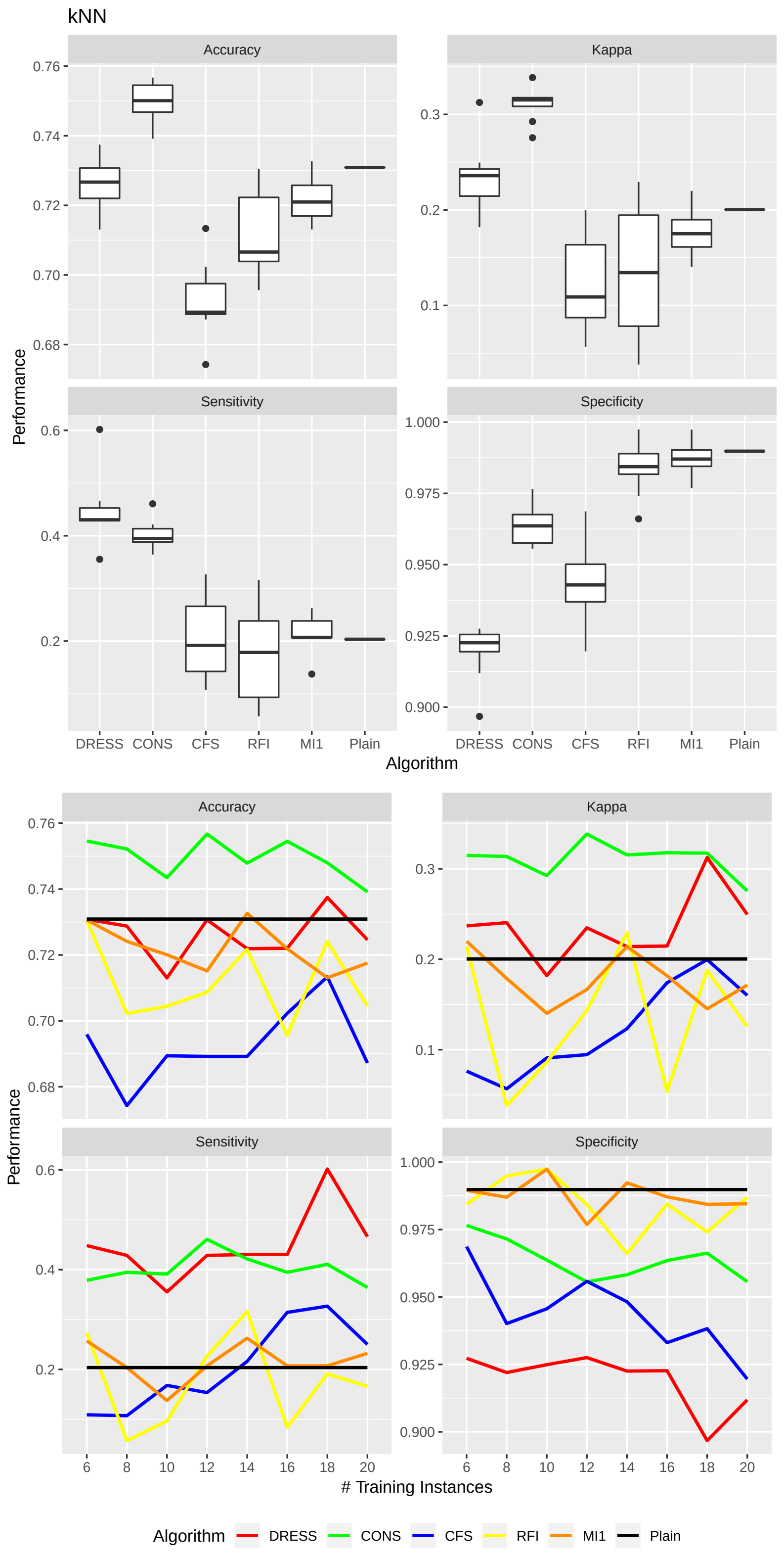


Figure 5.3: SHIP female partition kNN classification runs on feature subspaces.

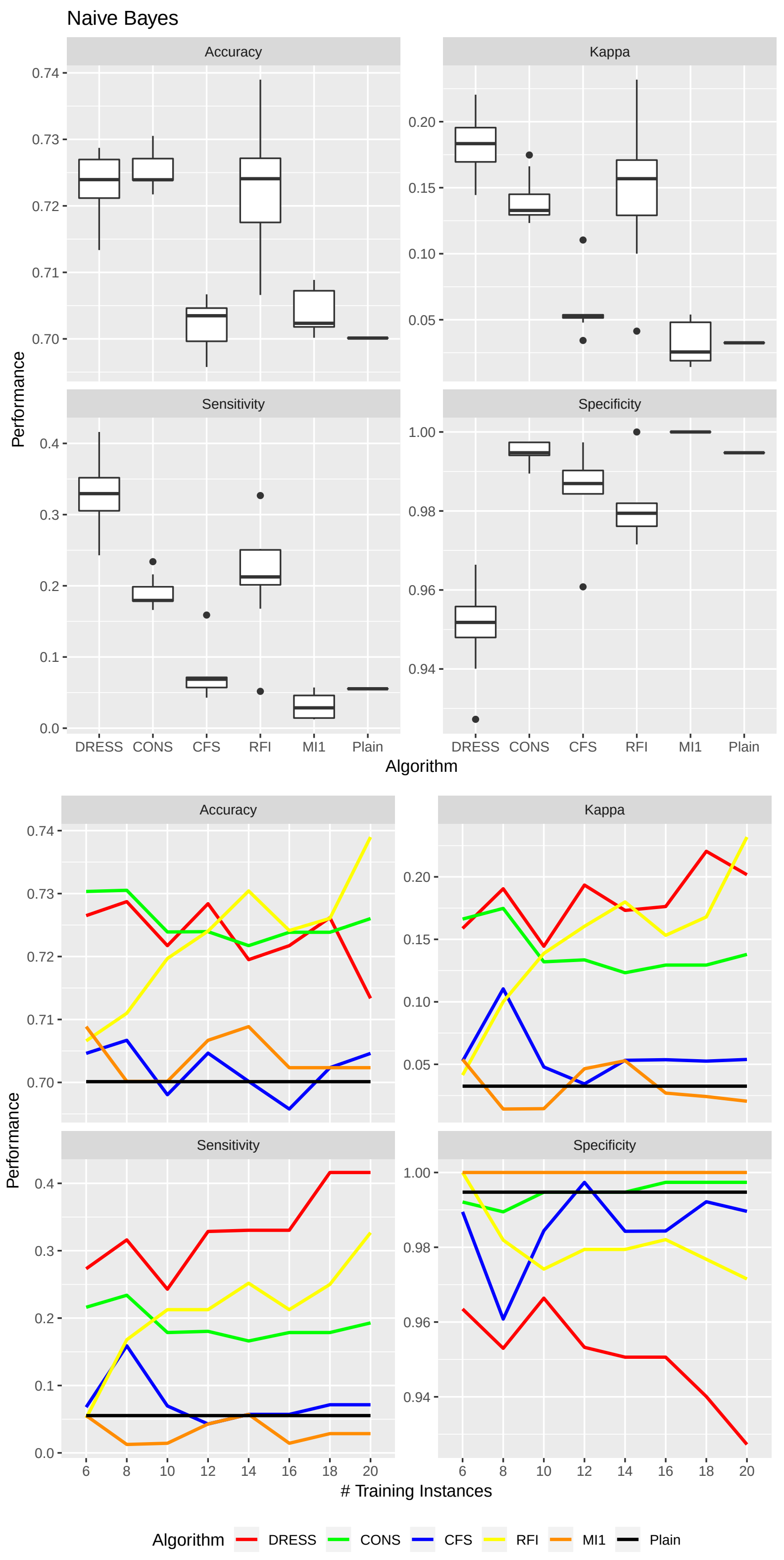


Figure 5.4: SHIP female partition Naive Bayes classification runs on feature subspaces.

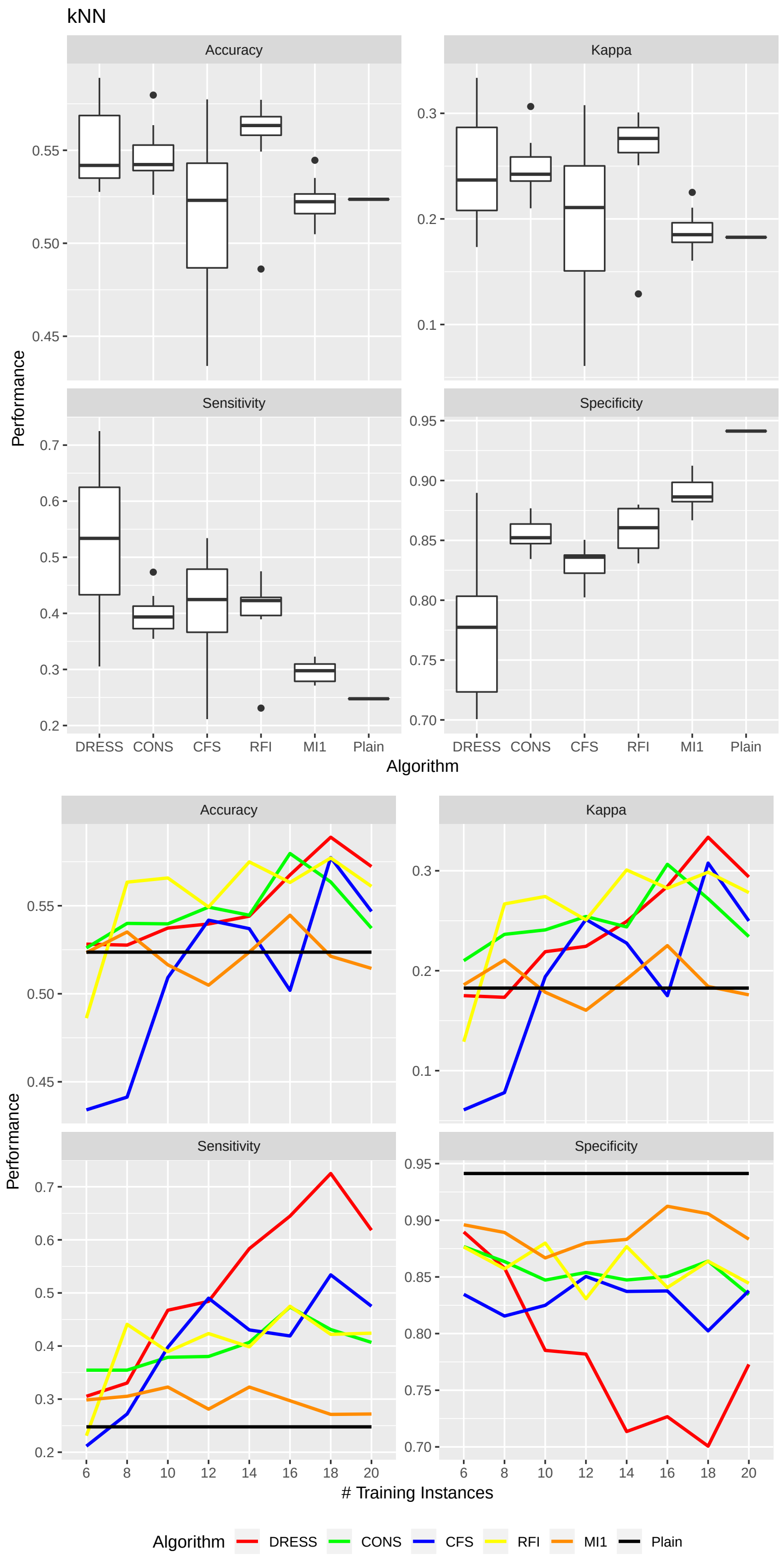


Figure 5.5: SHIP male partition kNN classification runs on feature subspaces.

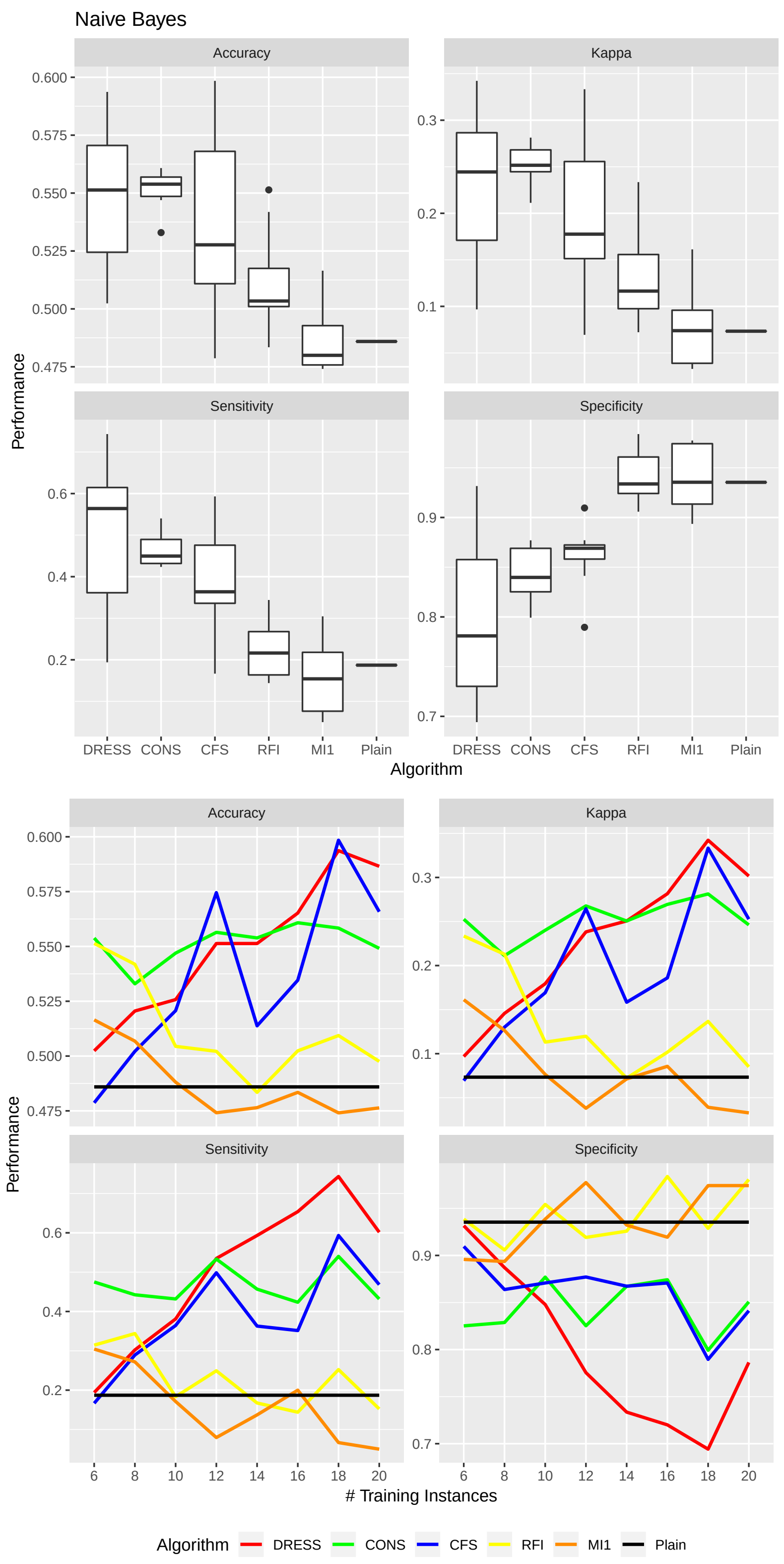


Figure 5.6: SHIP male partition Naive Bayes classification runs on feature subspaces.

For both classifier the performance of *DRESS* on the SHIP female partition is always among the highest for Kappa and Accuracy and clearly dominates the other variants on Sensitivity. Also, except for *DRESS*, there exists a clear gap between the order of the algorithms on the different performance measures, i.e. *CONS* clearly dominates *RFI* for the kNN variants and *RFI* dominates *CONS* on Naive Bayes, whereas *DRESS* always shows good performance. Considering that the class distribution is very skewed and the good Sensitivity results of *DRESS* (while still preserving a Specificity $\approx$ 90% - 97%), we conclude that it is the most balanced method w.r.t. the selection of features that are informative to both positive and negative instances.

Figure 5.5 and Figure 5.6 shows the Accuracy, Kappa, Sensitivity and Specificity line- and boxplots for all feature selection variants using the kNN (Figure 5.5) and Naive Bayes (Figure 5.6) classifier on the male partition of the SHIP dataset. Table 5.2 shows the average rank of each variant on the female partition of SHIP across all runs and classifiers. For kNN, *DRESS* achieves overall highest observed average Accuracy (58.8%), overall highest observed average Kappa (33%) and overall highest observed average Sensitivity (72.5%) among all variants when provided with 18 training instances. Generally, *DRESS* outperforms competitors w.r.t. Accuracy with 6,18 and 20 training instances, w.r.t. Kappa with 18 and 20 training instances and w.r.t. Sensitivity with 10,14,16,18 and 20 training instances. For Specificity, the Plain kNN always performs best with 94%, since it almost always predicts the majority class when using the complete feature space. However, DRESS never falls below 71.35% in Specificity. .

For Naive Bayes the subset of 18 training instances again leads to best performance values. Here, CFS achieves overall highest observed average Accuracy (59.84%) and DRESS achieves overall highest observed average Kappa (34.2%) and overall highest observed average Sensitivity (74.31%). RFI exhibits highest average Specificity (98.38%) with 16 training instances. *DRESS* outperforms competitors w.r.t. average Accuracy with 16 and 20 training instances, w.r.t. average Kappa with ,14,16,18 and 20 training instances and w.r.t. average Sensitivity with 12,14,16,18 and 20 training instances.

For the male SHIP partition, DRESS is the most balanced algorithm w.r.t. the selection of feature spaces that are informative regarding the positive and negative Hepatic Steatosis outcome. Over all provided training instance subsets, the boxplots show that DRESS has the second highest performance variance in Accuracy and Kappa (after CFS), and the highest performance variance in Sensitivity and Specificity. Together with the observation that the fully supervised CFS also performs better on the same training instance subsets where DRESS performs best, we conclude that DRESS is highly sensitive to the quality of the provided training instances benefiting/suffering more from good/bad instances than the competitors.

Avg. Rank	DRESS	CONS	CFS	RFI	MI1	Plain
Accuracy	2.625	1.4375	5.375	3.5625	4.0	4.0
Kappa	1.8125	1.8125	4.625	3.6875	4.6875	4.375
Sensitivity	1.125	2.25	4.3125	3.75	4.6875	4.875
Specificity	6.0	3.3125	4.4375	3.375	1.625	2.25

Table 5.1: Average rank of each variant across all classifiers on the female partition of SHIP.

Avg. Rank	DRESS	CONS	CFS	RFI	MI1	Plain
Accuracy	2.3125	2.125	3.75	2.8125	5.0625	4.9375
Kappa	2.375	2.0625	3.625	2.9375	4.9375	5.0625
Sensitivity	1.625	2.4375	3.125	3.625	4.9375	5.25
Specificity	5.375	4.1875	4.8125	2.875	2.25	1.5

Table 5.2: Average rank of each variant across all classifiers on the male partition of SHIP.

5.4.4 Evaluation on Subspace Cluster Validation

In this section we show findings and evaluate the transferability of DRESS results with the help of the validation component, i.e. we investigate whether identified subspace clusters can be discovered in an independent dataset. Additionally, we evaluate the impact of a variation of the DRESS constraints on the results and show relevant examples of subspaces and clusters. We utilize two real-world cohort study datasets on the medical conditions Hepatic Steatosis and Goiter to identify subspaces, show findings and assess their semantics.

	SHIP-2	**TREND-0**
Instances	1878	4400
Variables	70	492
Age [years]	58.1±13.5	51.0±14.1
Sex [% female]	53.3 %	51.7 %
Hep. Stea.	163 pos., 564 neg.	462 pos., 1464 neg.
Goiter	-	1390 pos., 3010 neg.

Table 5.3: Characteristics of the medical cohort datasets SHIP-2 and TREND-0.

5.4.5 Datasets and Matching for Subspace Cluster Validation

Since for the validation of DRESS's results independent datasets that contain comparable variables must exist, the datasets used of evaluation and the evaluation settings are adjusted. For the validation of subspace clusters and involved features we investigate results on real-world data from two comparable independent cohorts of the Study of Health in Pomerania (SHIP) that is conducted in Northeast Germany [93]. The subspace clusters were derived with the DRESS algorithm as presented in [31]. The first cohort we use is SHIP-2 and is described in the first part of Section 4.5.1. In addition we utilize a second cohort, (SHIP-TREND), that was established in the same region as SHIP-2. Here, 4,420 individuals aged between 20 and 84 participated in TREND-0 (the baseline examination). Table 5.3 presents general characteristics of SHIP-2 and TREND-0.

For these cohorts, we investigate the disorders hepatic steatosis (described in first part of Section 4.5.1) and goiter to explore the feasibility of our method. Presence of hepatic steatosis for study participants in both cohorts is indicated through a discretized variable based on the proportion of fat within the liver as measured through Magnetic Resonance Tomography (MRT). Liver-fat MRT results were available for 727 SHIP-2 and 1926 TREND-0 individuals. We use a binary variable "H" which marks participants with more than 10% of fat accumulation within their liver as "positive" and the remaining participants as "negative". The distribution within SHIP-2 and TREND-0 is presented in Table 5.4. For the prospensity score matching of both datasets we use a threshold of 0.05%

The second disorder we investigate is goiter. Goiter refers to an enlargement of the thyroid gland that may be defective. Prevalence is especially high in iodine-deficient regions of the world (up to 80 % [110]). The disorder of goiter is only present in TREND-0. Here, 1390 out of 4400 participants exhibit the disorder. For men, it is defined by a thyroid gland volume of more than 25 ml. For women, the gland volume must exceed 18 ml. The volume is measured by ultrasound, according to [111].

Propensity score matching as described in Section 5.3.5 is done for the disorder heaptic steatosis using the SHIP-2 dataset as DD and TREND-0 as VD. Since for goiter no independent dataset exists (it is only present in TREND-0), we conduct a 50/50 split of the TREND-0 dataset into a DD and VD set. The validation component matches the respective discovery and validation datasets according to their distribution on the following variables: age, sex and medical outcome, variables given by a domain expert. Further, in the following sections, DD and VD refers to the *matched* (when applicable) discovery and validation datasets. Table 5.4 displays the distribution of the datasets before and after matching.

5.4.6 Experimental Setup for Subspace Cluster Validation

For the subspace cluster validation we first investigate whether the identified subspace cluster also exist in independent, comparable datasets. In these experiments, DRESS discovers subspace clusters using five ML and five NL constraints that are chosen at random from the labeled data. We conduct three measurements for each subspace cluster C_{DD} discovered in DD and its reconstructed counterpart C_{VD} in VD as follows:

- p-value: p-value according to χ^2-test on the class distribution within the subspace clusters compared to the remaining instances in the respective dataset. We call a subspace cluster “significant” if its p-value is below the predefined level $\alpha = 0.05$. In the best case, subspace clusters that are found significant in DD should also be significant in VD.

- Relative Size Difference (RSD): measurement of the differences in relative size between a subspace cluster in the discovery set DD and its reconstructed counterpart in VD. RSD is calculated as

$$RSD(C_{DD}, C_{VD}) = \frac{abs(|C_{DD}| - |C_{VD}|)}{\frac{1}{2} \cdot (|C_{DD}| + |C_{VD}|)}. \tag{5.7}$$

  A low RSD indicates that subspace clusters have similar size on both datasets.

- Relative Class Distribution Difference (RCDD): measurement of the difference in the relative presence of the positive class in a subspace cluster in DD compared to its VD counterpart. The RCDD is calculated as

$$RCDD(C_{DD}, C_{VD}) = \frac{P(C_{DD})/P(DD)}{P(C_{VD})/P(VD)}. \tag{5.8}$$

  $P(C_{DD})$ is the fraction of instances with the positive class label in C_{DD}. $P(DD)$ is the fraction of instances with the positive class label in the complete DD. A low RCDD indicates that subspace clusters have a similar distribution w.r.t. the class variable on both datasets.

In the second part of the subspace cluster validation, we vary the number of constraints given to DRESS and investigate the changes in the proportion of significant clusters that are discovered. Here, we calculate the number of significant subspace clusters ($\alpha = 0.05$) in relation to all found subspace clusters, given a fixed number of random constraints.

Finally, in the last part of our experiments on subspace cluster validation, we show example subspace clusters and involved feature-values for both disorders when using DRESS on the combined DD and VD with 5 ML and 5 NL constraints, and assess the quality of our findings through a comparison with existing study results from independent literature.

5.4.7 Transferability of Identified Subspace Clusters

Figure 5.7 and Figure 5.9 present the median χ^2-test p-value of the 25 best subspace clusters identified in DD (left part of the figure), and the

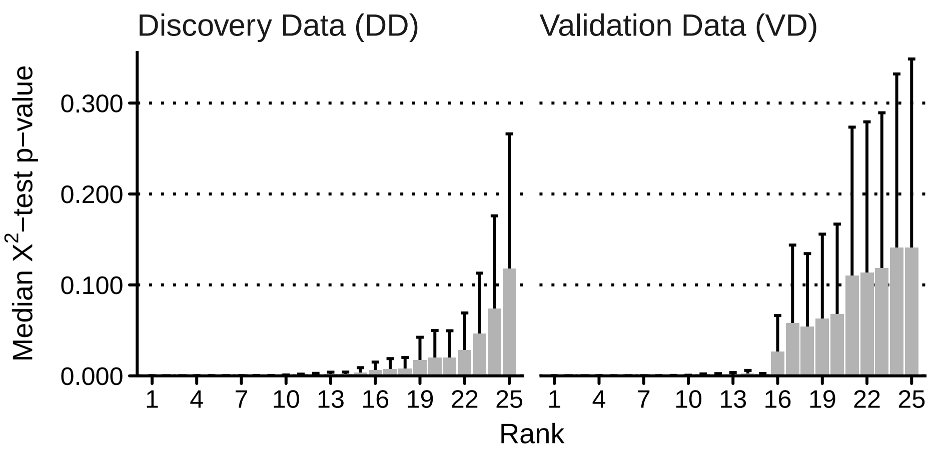


Figure 5.7: χ^2-test p-value median ranking over 20 runs for subspace clusters found in the hepatic steatosis DD compared to the VD counterpart. Ordering of the subspace clusters in DD is mostly preserved in VD. P-values are mostly comparable.

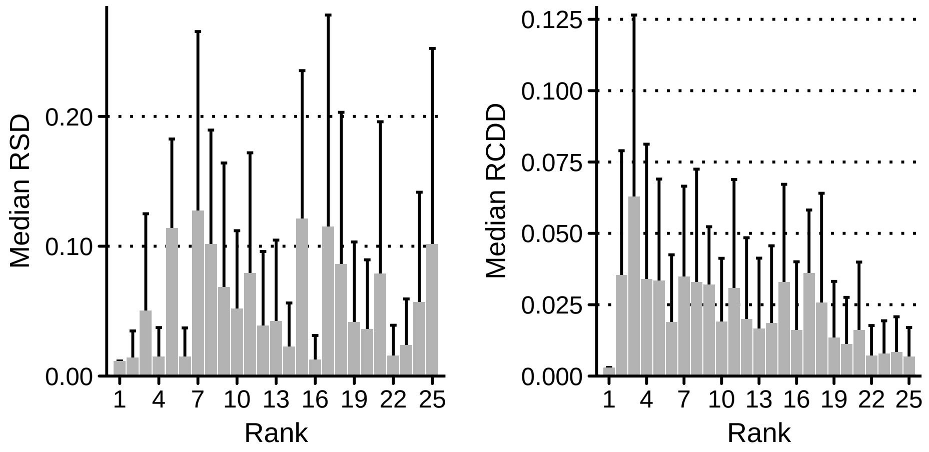


Figure 5.8: Median RSD and RCDD for the 25 best ranked subspace clusters found in the hepatic steatosis dataset.

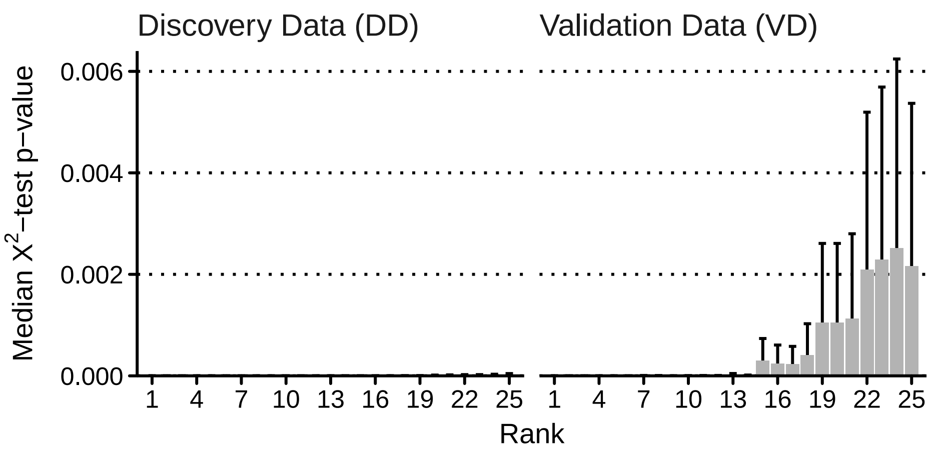


Figure 5.9: χ^2-test p-value median ranking over 20 runs for subspace clusters found in the goiter DD compared to the VD counterpart. Ordering of subspace clusters in DD is not completely preserved in VD. However, p-values are very low and similar for all subspace clusters.

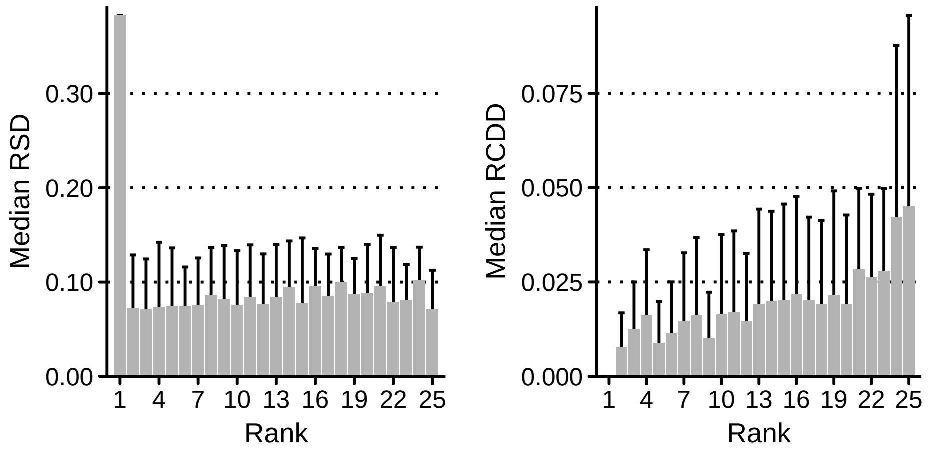


Figure 5.10: Median RSD and RCDD for the 25 best ranked subspace clusters found in the goiter dataset.

	SHIP-2	**TREND-0**	**p-value**
	BEFORE matching		
n	727	1926	-
Age [years]	56.1±12.6	50.1±14.1	<0.001
Sex [% female]	53.2 %	51.7 %	0.512
Hep. Stea. [% pos.]	22.4 %	24.0 %	0.426
	AFTER matching		
n	694	694	-
Age [years]	55.5±12.1	55.5±12.1	1.000
Sex [% female]	53.5 %	53.5 %	1.000
Hep. Stea. [% pos.]	21.5 %	21.5 %	1.000

Table 5.4: Matching of the hepatic steatosis datasets SHIP-2 and TREND-0.

associated subspace clusters in VD (right part of the figure), for hepatic steatosis and goiter over 20 runs. For hepatic steatosis, the p-value between subspace clusters in DD and their VD counterpart are highly similar up to subspace cluster rank 16. Also, up to rank 16, subspace clusters are significant. Subspace clusters from rank 23 to 25 are not significant in both datasets. Therefore, the majority of the high-quality subspace clusters are transferable. It is observed that the subspace clusters between rank 17 and 22 are significant in DD but not significant within VD. However, in many instances these ranks were close to the decision border (i.e., 17, 18, 19, 20). Another favorable result comes from the order of the subspace clusters. With the exception of rank 14 and 18, the ordering is mainly preserved in VD, thus indicating stable results of STATIC-Select. In Figure 5.8 we present RSD and RCDD on the hepatic steatosis data. RSD shows some variability but is generally low, exhibting a median of $\approx 12\%$ in the worst and $\approx 1\%$ in the best cases. When a DD subspace cluster is large, a RSD of 12% is correspondent to an approximate size difference of 15 instances between the DD cluster and its VD counterpart. We also observe a low RCDD across all subspace clusters (median values approx. between 1.5% and 3.5%).

The picture changes slightly for goiter. All 25 subspace clusters are highly significant on both DD and VD. P-values increase slightly from rank 15 onwards on VD. However, they remain considerably below 0.01. We further observe that the ordering is not fully preserved in VD, though the absolute deviation in p-value is very low, especially in comparison to the hepatic steatosis data. Figure 5.10 shows the median and absolute median deviation of the RSD and RCDD. With the exception of the best ranked subspace clusters, the median RSD is low and never exceeds $\approx 10\%$. Additionally, there is no medical outcome distribution difference for the best ranked subspace clusters. Finally, RCDD is generally very low. The median never exceeds $\approx 5\%$, thus indicating good transferability of discovered subspace clusters.

5.4.8 Varying the Number of Constraints

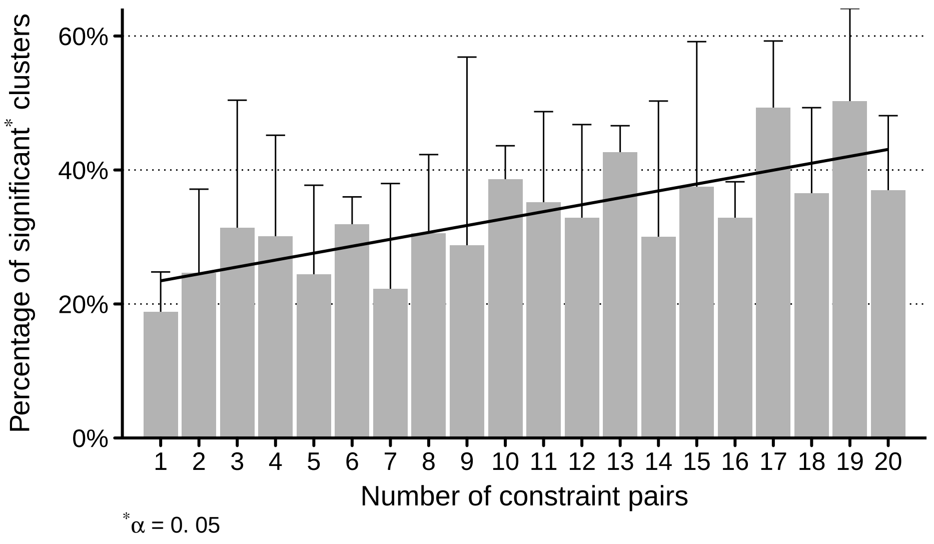


Figure 5.11: Percentage of significant subspace clusters when varying the number of constraint pairs.

In Figure 5.11, the relative proportion of significant subspace clusters ($\alpha = 0.05$) when varying the number of constraint pairs on the matched SHIP-2 DD is shown. Again, the constraint pairs were chosen at random, (with an equal number of ML and NL constraints) and were varied from one

Hepatic Steatosis				
ID	**Subspace features**	**Size [%]**	**p-value**	**OR**
H#1	`age`,`diabetes`	10.0	$1.7\,\mathrm{e}^{-09}$	3.01
H#2	`female`,`smoking`	11.0	$5.4\,\mathrm{e}^{-09}$	0.31
H#3	`abstain`,`physact`,`smoking`	9.9	$2.3\,\mathrm{e}^{-09}$	2.06
H#4	`ATC_C07A`	21.9	$2.7\,\mathrm{e}^{-09}$	2.70

Goiter				
ID	**Subspace features**	**Size [%]**	**p-value**	**OR**
G#1	`ges_sf12_02`,`waiidf`	33.6	$1.8\,\mathrm{e}^{-09}$	0.47
G#2	`edyrs`,`metsyn`	26.9	$7.\,\mathrm{e}^{-09}$	1.55
G#3	`ges_sf12_03`,`plaque`	16.0	$3.4\,\mathrm{e}^{-09}$	1.60
G#4	`ffs`,`marit`	4.4	$8.2\,\mathrm{e}^{-09}$	2.07

Table 5.5: Description and statistics of selected subspace clusters.

to 20 ML and NL constraints. With each constraint pair, ten runs were conducted and the associated median and median absolute deviation was calculated. Figure 5.11 indicates an increase in the relative proportion of significant subspace clusters when the number of constraints is increased. For the median measurements, we observe some variability in the data. Median relative proportion of significant subspace clusters for one ML and NL constraint is $\approx 20\%$, has its maximum at $\approx 47\%$ (with 19 pairs of constraints) and settles at $\approx 38\%$ (with 20 pairs of constraints). In contrast, the median absolute deviation is moderately high but constant over the tested constraint pairs. This may be due to the randomness of the chosen constraint pairs. As described, the constraints selection here is solely based on the class-similarity of instances and does not take their similarity w.r.t. specific features into account. This may lead to the selection of constraints that not reflect the underlying concept well, for example ML constraints between highly dissimilar outlier instances that have the same class but are not representative. An expert with domain knowledge (if available) that defines a number of constraints manually is likely to take more information than the class label into consideration when deciding on constraints.

5.4.9 Relevant Example Subspaces

Table 5.5 depicts eight selected subspace clusters with significantly different class distribution, in comparison to all instances of the dataset, for the outcome hepatic steatosis and goiter. All of the eight depicted subspace clusters are significant ($\alpha = 0.01$) and exhibit a high odds ratio. In Figure 5.12 (hepatic steatosis) and Figure 5.13 (goiter) we depict the class distribution of these subspace clusters as well as the clusters distributions of the included features in comparison to the overall dataset. Table 5.6 provides a brief description of each feature contained in an identified subspace cluster.

Example Subspace Clusters on Hepatic Steatosis

The instances in subspace cluster H#1 exhibit higher age, in comparison to the remaining study participants, and the diabetes condition. These participants have a high odds ratio and are almost equally distributed (which is considerably different to the complete cohort) between the positive and negative hepatic steatosis outcome. Publications such as [112] and [113] support this observation. For H#2, included participants are females, current smokers and have low odds ratio. On the other hand, the participants in H#3 are ex-smokers that were not abstinent from alcohol in a time-frame of one year and which perform less than one hour of sports activities per week. Correlated features to those identified in H#2 and H#3 (like body-mass-index or waist circumference) were also described as being important w.r.t. hepatic steatosis in the studies [27] and [29]. Finally, the subspace cluster H#4 contains participants under beta-blocker medication. A hint on the relevance of this characteristic regarding hepatic steatosis is found in [114]. Here, a diagnostic score on fatty liver uses this information on some occasions.

Example Subspace Clusters on Goiter

In the lower part of Table 5.5 we present statistics and contained features of selected subspace clusters on goiter. G#1 contains mainly negative goiter outcome instances that are free from impairments for moderate physical tasks (`ges_sf12_02` $= 3$) and without central obesity as categorized by the International Diabetes Federation (IDF). Instances of G#2 are associated with (relative) few years of education (compared to the complete cohort)

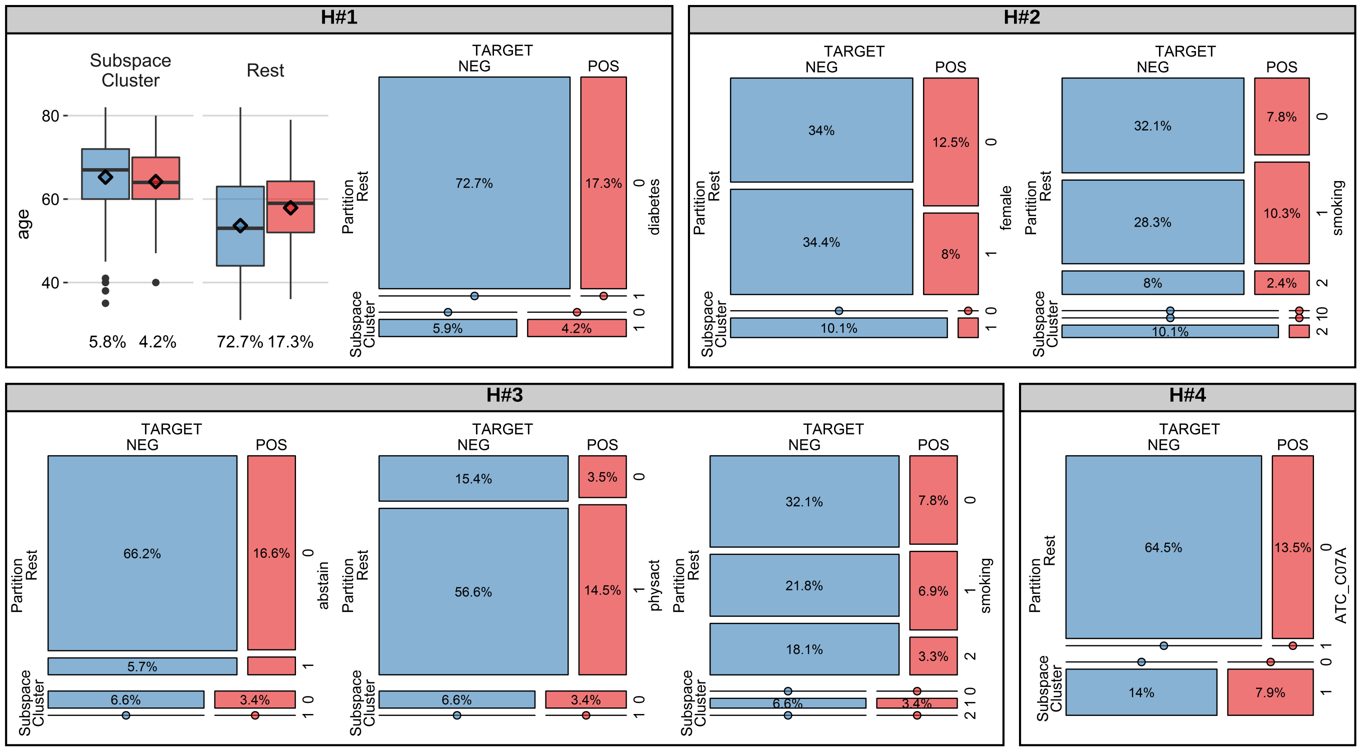


Figure 5.12: Class- and feature-distribution w.r.t. hepatic steatosis for subspace clusters identified by DRESS. Box-plots for independent continuous variables and mosaic charts for independent nominal variables. Color indicates class membership. subspace cluster H#1 represents approx. 10% of the dataset's instances. Those instances exhibit a higher relative proportion of a positive hepatic steatosis outcome and refer to considerably older and diabetes-afflicted study-participants. H#2 contains females that are currently smokers. Both feature-values in this cluster are associated with a higher proportion of negative-class participants. subspace cluster H#3 again exhibits a higher relative proportion of the positive hepatic steatosis class. This cluster contains study participant that are ex-smokers, less alcohol-abstinent and less physically active than the remaining study participants. H#4 Contains participants under beta-blocker medication which generally are more likely to exhibit the positive hepatic steatosis outcome.

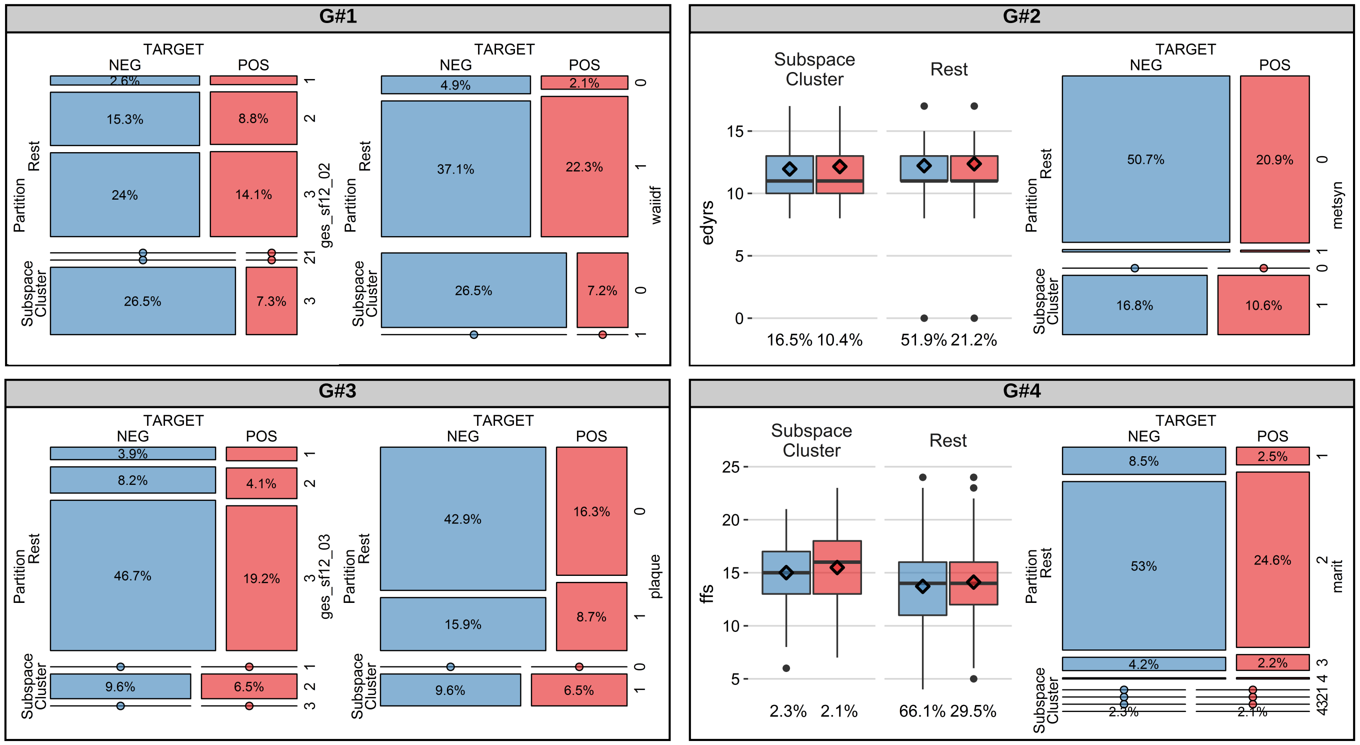


Figure 5.13: Class- and feature-distribution w.r.t. goiter for subspace clusters identified by DRESS. Box-plots for independent continuous variables and mosaic charts for independent nominal variables. Color indicates class membership. Instances of subspace cluster G#4, account for 4.4% of the study participant and exhibit a higher relative proportion of a positive goiter outcome. They are are widowed and adhere to a more favorable diet than the remaining study participants. In contrast, the instances of G#1 (which contains 33.6% of the study participant population) exhibit a higher relative proportion of a negative goiter outcome. Those participants have a normal waist circumference but also exhibit severe limitations while performing moderate physical activities (i.e. moving a table, vacuuming, bowling). G#2 and G#3 are associated with a higher relative proportion of the positive goiter outcome. Here, G#2 participants have a higher variance in the number of educational years and exhibit the metabolic syndrome, whereas G#3 participants exhibit plaque and report difficulties when performing physical tasks.

and suffer from the metabolic syndrome. This cluster shows a distinctively high proportion of the positive goiter outcome. Different studies support this, as they show important associations between the educational level, somatographic variables and the metabolic syndrome with goiter [115], [116]. In contrast, our found associations between goiter (positive outcome) and presence of plaque `G#3` have not yet been confirmed in literature. On the other hand, `ges_sf12_02` which is also part of `G#3`, shows high correlation to the somatographic variables. At last, `G#4` contains participants with high food frequency score (i.e. participants that adhere to a more recommended diet) which are widowed. This cluster exhibits a higher relative proportion of the positive goiter outcome than the rest of the cohort. These associations remain to be confirmed in independent studies, however they may be suitable to formulate new hypothesis on.

5.5 Conclusions on Methods

Methods that aim to identify relevant dimensions for objects suffer when only limited ground-truths is available. In many real-world domains, the amount of labeled data is sparse and the process of labeling is long and expensive due to domain expert involvement. It is therefore mandatory to develop algorithms that can cope with exploiting the limited available knowledge and still produce satisfactory results. We propose a module that uses traditional feature selection when labeled training data is widely available and a novel constraint-based subspace clustering algorithm DRESS (Discovery of Relevant Example-constrained SubSpaces) when not. The traditional feature selection approach is based on CFS [12] where subsets of features with the least redundancy among each other and a maximum relevance w.r.t. the target variable are selected with the help of a greedy forward-selection. On the other hand, our algorithm DRESS uses instance-level similarity constraints to evaluate feature subspaces on constraint satisfaction (with the help of density-based clustering) and the subspace-specific similarity of instances under constraints. The used constraints reflect implicit knowledge on the instances w.r.t. their similarity on the target concept. They can be either derived from class-labels or manually assigned by domain experts. Connected to DRESS, is a component that transfers subspaces and clusters on an independent dataset to validate their relevance in order to check the ability of DRESS to produce generalizable results. In our experiments we evaluated the quality of subspaces produced by DRESS for two datasets and classifiers against the feature subspaces produced by traditional feature selection algorithms. The results show that DRESS produces the most balanced results w.r.t. the prediction of the positive and negative outcome, ranks among the top algorithms on Accuracy, Kappa and Sensitivity and identifies feature subspaces that substantially offset the baseline classifiers bias in predicting the dominant class.

Further, with the help of the validation component we evaluated whether DRESS produces significant and generalizable results by transferring the produced subspaces and clusters of two medical real-world datasets on a second validation dataset. Results show that a majority of the subspace clusters with high significance are confirmed on the independent dataset for both investigated classification tasks. The framework identifies highly significant clusters and successfully reconstructs them in the respective validation datasets retaining their significance in the process. Also, it is shown that an increase in the given constraints for DRESS leads to the identification of a higher proportion of significant subspace clusters. A random selection of no more than 3% of the labeled data was sufficient to identify subspace clusters that were shown to be relevant in independent studies.

The results achieved by DRESS w.r.t. traditional feature selection algorithms indicate that a semi-supervised approach in conjunction with an alternative type of background knowledge improves the quality of selected feature subspaces for classification tasks when ground-truth is sparse. Therefore, we answer RQ 2 positively.

One limitation of STATIC-Select is the fact that the switch between traditional method and DRESS is hard to make before any tests on the specific dataset are made. This is the case when a semi-high amount of labeled training data is available and it is unclear whether a traditional algorithm or DRESS is the better option. Therefore, we suggest a prior evaluation on parts of the datasets in question which involve both options. Further, DRESS is currently not directly applicable on the historical records of instances. Instead, DRESS requires the preprocessing of the EVO-Extract module to handle evolving objects.

With respect to the validation module, the proposed mechanisms can only be utilized if comparable datasets with similar classification problems are available. In particular, for this thesis, a second independent dataset was only available for hepatic steatosis. For goiter, we had to reserve a part of TREND-0 for the validation, rather than use the SHIP-CORE, because the major differences in covariates between the two cohorts did not allow for a matching.

In the future, we will enhance DRESS to exploit the temporal dimension by defining the similarity of instances based on their previous evolution. This removes a preprocessing step and allows experts to find groups of similar evolving instances directly, at the cost of understandability of the involved features. Another future prospect is the shift from a global view on relevant subspaces by means of subspace scoring to a local cluster-based view. Instead of accessing the relevance of subspaces and extracting all clusters from it, we want to consider each cluster-subspace pair individually and combine hierarchical with constraint-based subspace clustering. This could not only lead to the discovery of subspaces relevant w.r.t. a specific class, but also to the identification of distinct groups and associated features within subclasses/subconcepts.

name	**description**	**values**
`abstain`	Abstinence from alcohol (12 months)	0: no 1: yes
`age`	Age of participant	*numeric*
`ATC_C07A`	Intake of beta-blocker	0: no 1: yes
`diabetes`	Suffers from diabetes	0: no 1: yes
`edyrs`	Number of educational years	*numeric*
`ffs`	Food Frequency Score	*numeric*
`ges_sf12_02`	Impaired performing moderate tasks	1: severe limitation 2: slight limitation 3: no limitation
`ges_sf12_03`	Impaired walking multiple stairs	1: severe limitation 2: slight limitation 3: no limitation
`marit`	Marital status	1: single 2: married or relationship 3: separated or divorced 4: widowed
`metsyn`	Suffers from metabolic syndrome	0: no 1: yes
`physact`	Leisure time physical activity	0: $<$ 1h phys. act./week 1: $\geq$1h phys. act./week
`plaque`	Plaque	0: no | 1: yes
`smoking`	Smoking status	0: never smoked 1: ex-smoker 2: current smoker
`waiidf`	Waist circumference	0: $<$ 80 cm (men: 94 cm) 1: $\geq$ 80 cm (men: 94 cm)

Table 5.6: Description of selected features. Box-plots for independent continuous variables and mosaic charts for independent nominal variables. Color indicates class membership.

6. Conclusion & Future Work

In this thesis we proposed a framework for the identification on relevant dimensions for evolving objects based on different types of background knowledge. Traditional systems suffer from obscure representations of evolving objects, failing to make the implicit dimensionality of an object's evolution explicit, and the requirement for large amounts of labeled data to select relevant subsets of features. We argue that it is beneficial to represent such objects based on their historical records and select relevant features based on an evaluation of the internal structure of feature subspaces guided by limited knowledge on the similarity of instances in those subspaces. In this context, we proposed a framework consisting of two distinct components for the extraction and selection of relevant dimensions and formulated associated research questions that are addressed in the subsections below. Furthermore, in this chapter we answer our core research question and draw final conclusions from the combined contributions in this thesis. Finally, limitations of our proposed approaches and directions for future research are discussed.

6.1 Extracting Features from Object Evolution

The first component of this thesis and our framework is EVO-Extract which aims to codify implicit dimensionality given by the historical records of evolving objects by generating features from the observed evolution, that are relevant regarding the target concept as given by a classification problem. This component is related to the first research question:

RQ 1: How can we codify the implicit dimensionality of an object's historical records to improve object classification?

To answer this question and derived subquestions we proposed two general approaches that derive so called Evolution Features from the evolution of objects: We developed a method that takes as input any time-series feature with a discrete temporal dimension to automatically find and generalize sequential patterns relevant to the target-concept. These groups of patterns are then used to generate new features based on the objects pattern group-memberships. The second method we proposed tracks objects and object-groups in relevant subspaces over the temporal dimension and derives pre-defined features based on the change in the position of each object and its peers.

In our evaluation we use ten classification algorithms and compare their performance on a classification problem when using the original feature set, the derived Evolution Features, a number of simple statistics derived from the object evolution and each possible combination of the aforementioned three variants in a 10-fold cross-validation. We further vary the use of a prior feature selection for each variant, which equals 4800 evaluation runs in total. As evaluation measures we calculate Accuracy, Kappa, Sensitivity, Specificity and a rank based on the mean over all measures.

Our experimental results on four heterogeneous real-world datasets show, that in each case the best rank was achieved by a classification algorithm that utilizes the derived Evolution Features. Furthermore, depending on the dataset, between 75% and 100% of all classification algorithms that used a feature set with Evolution Features performed better (as measured through rank) than their counterparts that did not utilize the same. This allows us to validate the following inequality (cf. Section 4.2):

$$perf(model(D_{tr}, F^* \cup G^* \cup G^*_{evo}), D_{te}) > perf(model(D_{tr}, F^* \cup G^*), D_{te}) \tag{6.1}$$

Our results indicate that for at least 75% of tested classification algorithms and for 100% of the overall top-performing algorithms, the derived Evolution Features contribute to a class separation. Therefore, we answer RQ 1 positively with the proposed methods.

This answer has implications for domain experts and the application of classification methods in real-world domains. Our findings show that the representation of evolving objects w.r.t. a target concept must be carefully planned and investigated due to the high impact on classification performance and the relevant knowledge that is hidden in the object evolution. With such representation techniques, it is possible to provide new insights for experts and to enable the formulation of new hypothesis regarding the concept at hand (for example new hypothesis on the onset of disorders or diseases in medicine).

6.2 Selecting Relevant Features

The second component of this thesis and our framework is STATIC-Select which aims to identify relevant features (w.r.t. the target concept) from the set of dimensions that are used to describe the objects investigated in this thesis. This component is related to the second research question:

RQ 2: How to identify the relevant features of evolving objects by exploiting background knowledge?

To answer this question we proposed a method that takes as input a dataset of static objects and a set of instance-level constraints to identify a subset of relevant features without the necessity to set any parameters. Our proposed method is especially well suited when only limited background knowledge exists, it can either derive similarity/dissimilarity constraints from a small set of labeled training data or the constraints can be given directly.

In our evaluation we use two naive classifier (kNN and Naive Bayes) and compare their performance on a classification problem when using the feature sets of our algorithm $DRESS$, the feature sets derived by four competing supervised and semi-supervised feature selection methods as well as with the complete original feature set in a 10-fold cross validation with 6,8,10,12,14,16,18 and 20 random training instances. As evaluation measures we calculate Accuracy, Kappa, Sensitivity and Specificity. We further show examples of derived feature subspaces for two disorders (hepatic steatosis and goiter), validate the subspaces on an independent cohort dataset and assess whether the identified features have known associations with the target outcome.

Our experimental results show that the feature subspaces discovered by DRESS improve the base classifier substantially. DRESS is able to outperform most competitors on each dataset and for all classifiers w.r.t. Accuracy and Kappa, and is the dominant algorithm for the identification of feature subspaces that are relevant towards the minority class. Furthermore, the validation component showed that the clusters used for feature subspace evaluation that were identified by DRESS are general w.r.t. the target concept in the sense that they can be reconstructed on a completely independent dataset where they exhibit similar size and distinct class distribution. This allows us to validate the following inequality (cf. Section 5.1):

$$perf(model(D_{tr}, F^*), D_{te}) > perf(model(D_{tr}, F), D_{te}) \tag{6.2}$$

Since our results indicate that our proposed algorithm $DRESS$ both selects feature subspaces that improve the performance of base classifiers (while also outperforming most competitors) and discovers feature subspaces that have known associations for two medical outcomes, we conclude that $DRESS$ identifies feature subspaces relevant to the target concept. Therefore, we answer RQ 2 positively with the proposed method. This answer has implications for real-world classification domains where ground-truth is scarce or unavailable. Our findings show that small sets of labeled training data or instance-level similarity constraints can lead to the identification of relevant features w.r.t. the target concept. Therefore, practitioners and experts are able to gain domain insights with a fraction of the general expenses (time, money, workforce etc.) that are normally required for dataset labeling.

6.3 Core Research Question

The core research question that was defined in Section 1.2, is:

How to find relevant dimensions of evolving objects for classification tasks with the help of background knowledge?

In Section 2.3 we presented the formalism to determine whether our proposed framework answers the question positively or not, which included the following expression (Expression 3.4):

$$expr(RQ1) \lor expr(RQ2)$$

We consider the core research question as *"positively answered"* if, and only if, the logical expression 3.4 holds true, and if the associated component(s) that led to this result utilized at least two different types of background knowledge to achieve this. We therefore investigate the different cases that emerge from these considerations: According to Section 2.3, $expr(RQ1)$ is true when RQ 1 is positively answered, i.e. when Inequality 4.1 holds true so that EVO-Extract derives features from the evolution of objects that improve the classification performance on the same. Given that this is the case (cf. Section 6.1), we consider $expr(RQ1)$ as true. According to Section 2.3, $expr(RQ2)$ is true when RQ 2 is positively answered, i.e. when Inequality 5.1 holds true so that STATIC-Select identifies features from the given set of explicit dimensions that improve the classification performance on the objects. Given that this is the case (cf. Section 6.2), we consider $expr(RQ2)$ as true. Since the core research question also expects that different types of background knowledge are exploited to answer the sub-questions positively, we next investigate the different types of knowledge used. In EVO-Extract, we use the historical records of objects and class-labeled training data to derive Evolution Features that successfully improve classification performance. Therefore, EVO-Extract successfully exploits two different types of background knowledge. In STATIC-Select, we use either class-labeled training data or similarity constraints to successfully decide on relevant feature subsets for classification tasks. Therefore, STATIC-Select successfully exploits two different types of background knowledge.

Considering that $expr(RQ1)$ and $expr(RQ2)$ are true and both respective framework components successfully use two different types of background knowledge (three different types in total) to achieve this, it follows that we consider the core research question as positively answered.

This conclusion has consequences for practitioners and domain experts. In many domains, objects are investigated with a specific target concept in mind. For example, medical experts seek to identify important characteristics associated with a disease or stock investors want to discover indicators to outperform the general market. In these cases, multiple challenges exist especially in the context of real-world data: Objects are *not* sanitized or adhere to strict assumptions. Instead, they are described by a plethora of differently scaled features which are not all relevant but may also contain irrelevant information regarding a target. A second layer of complexity is added by the fact that such objects are not static but may also evolve over time, containing implicit relevant information in their evolution. Therefore, to provide experts with relevant knowledge, both of these challenges must be investigated in conjunction. In this thesis we proposed a framework that can be used to derive information from object evolution and identify the information that is relevant w.r.t. the target concept. Experts can exploit the derived information to gain a deeper domain understanding and base new hypothesis on these insights. With the exploitation of similarity constraints for the identification of relevant dimensions, we also provide experts the opportunity to gain domain insights even when traditional ground-truth is scarce, unavailable or undesired. Instead, experts can input constraints between objects that they deem as being similar or dissimilar w.r.t. the experts target concept, to find out which dimensions actually constitute this similarity.

6.4 Limitations

Both main modules of our framework are evaluated on real-world datasets with an associated real-world classification problem. These datasets come from a medical domain. In contrast, a number of different domains with different datasets may exist where our framework could be applied onto. It is not clear how our findings would transfer to those domains. However, the real-world datasets we utilize are much more complex than the typical arbitrary datasets used for evaluation. The components of our framework are evaluated on objects that are described by differently scaled features that can contain missing values, irrelevant and redundant information. The difficulty of the data we use, together with the strict evaluation protocol, indicate that our proposed methods perform well.

Furthermore, several methods that share similarities to ours exist that were not included in the evaluation, since they have some weaknesses for an application in real-world settings (for example the number of parameters or an unclear methodology). It is possible that these methods can outperform the presented ones under the right circumstances and with the right parameter settings. However, we concentrated to develop methods that do **not** require any parameter optimization and are applicable by domain experts without a deep understanding w.r.t. the underlying algorithms, which can easily shift out of focus when developing methods that solely concentrate on a theoretical sound foundation.

At last the application of significance tests was only conducted for STATIC-Select as a means to show if subspace clusters found by DRESS are significantly different in their class distribution compared to the overall population (cf. Section 5.4.6). This showed that the features associated with these subspace clusters are indeed informative regarding different labels. We omitted tests for significance in Chapter 4 since our main interest was to investigate whether Evolution Features improve classifiers. Results showed that Evolution Feature variants clearly dominated their counterparts and the top-25 variants (out of 120 variants) by large margins.

6.5 Future Work

During the work on the components of our framework we recognized several ways for a future expansion. Since the generation of additional Evolution Features does not inhibit the classification performance in the context of the framework (irrelevant and redundant features will be eliminated), EVO-Extract may be horizontally and vertically extended. Vertically, the proposed methods may incorporate additional pre-defined features that can be cluster- or instance-based. Horizontally, completely different methods that cater to different objects may be included. For example, features that are based on additional temporal discretization (in the case of a continuous temporal dimension) like obtained with KARMA-LEGO [77, 117]. For STATIC-Select, especially the extension of DRESS is promising. DRESS has already shown that it can beat competing feature selection algorithms, w.r.t. to the selection of features for a classification problem in a static scenario, when ground-truth is sparse. However, especially for evolving objects with time-series features that have a continuous temporal dimension, a direct applicability of DRESS on such objects may be beneficial. In this case, the discretization of the temporal dimension could be omitted and thus any associated information loss w.r.t. the target concept is prevented. To achieve this, we must investigate how an adequate notion of similarity between such objects can be defined and how the resulting feature subspace (that may consist of both static and time-series variables with a continuous temporal dimension) can be returned in a way that is insightful for the domain expert. Also, an investigation concerning which classification methods benefit the most from both EVO-Extract and STATIC-Select could maximize the benefit gained with both components. In this case, the comparison between baselines and enhanced classifiers should involve testing of significance in the case of small margins.

Further, the current implementation of our framework allows for many different modifications that are worth investigating: With respect to the proposed methods in EVO-Extract, the incorporation of alternative clustering algorithms and quality measures could lead to a boost in classification performance through the identification of better sequence-patterns. On the other hand, the DRESS algorithm presented in STATIC-Select can be modified w.r.t. the subspace search heuristic, the clustering algorithm as well as the internal quality function that incorporates constraint satisfaction and object distance. For example, instead of exploiting the pairwise distance between objects under constraints, we could utilize the Laplacian score to estimate how well locality is preserved by feature subspaces for constrained objects [118]. Also, additional methods to transform types of background knowledge could lead to the application of methods on datasets where these cannot currently be applied. For example, if for a dataset with evolving objects only a number of constraints is given, techniques that creates dummy labels out of those constraints could enable the application of the complete EVO-Extract component for this dataset.

Bibliography

[1] S. L. Cichosz, M. D. Johansen, S. T. Knudsen, T. K. Hansen, and O. Hejlesen, "A classification model for predicting eye disease in newly diagnosed people with type 2 diabetes," *Diabetes research and clinical practice*, vol. 108, no. 2, pp. 210–215, 2015. (cited on Page 1)

[2] S. Aich, K. Younga, K. L. Hui, A. A. Al-Absi, and M. Sain, "A nonlinear decision tree based classification approach to predict the parkinson's disease using different feature sets of voice data," in *20th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2018, pp. 638–642. (cited on Page)

[3] S. Vijayarani and S. Dhayanand, "Data mining classification algorithms for kidney disease prediction," *International Journal on Cybernetics and Informatics (IJCI)*, 2015. (cited on Page 1)

[4] M. F. Mirabelli, D. R. Ifa, G. Sindona, and A. Tagarelli, "Analysis of sexual assault evidence: statistical classification of condoms by ambient mass spectrometry," *Journal of Mass Spectrometry*, vol. 50, no. 5, pp. 749–755, 2015. (cited on Page 1)

[5] M. Kayser, "Forensic dna phenotyping: predicting human appearance from crime scene material for investigative purposes," *Forensic Science International: Genetics*, vol. 18, pp. 33–48, 2015. (cited on Page)

[6] B. Kong, D. Ramanan, and C. Fowlkes, "Cross-domain forensic shoeprint matching," in *British Machine Vision Conference (BMVC 2017)*, 2017. (cited on Page 1)

[7] V. Van Vlasselaer, C. Bravo, O. Caelen, T. Eliassi-Rad, L. Akoglu, M. Snoeck, and B. Baesens, "Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions," *Decision Support Systems*, vol. 75, pp. 38–48, 2015. (cited on Page 1)

[8] P. Save, P. Tiwarekar, K. N. Jain, and N. Mahyavanshi, "A novel idea for credit card fraud detection using decision tree," *International Journal of Computer Applications*, vol. 161, no. 13, 2017. (cited on Page)

[9] E. Duman and Y. Sahin, "A comparison of classification models on credit card fraud detection with respect to cost-based performance metrics," *Use of Risk Analysis in Computer-Aided Persuasion. NATO Science for Peace and Security Series E: Human and Societal Dynamics*, vol. 88, pp. 88–99, 2016. (cited on Page 1)

[10] U. Niemann, H. Völzke, J.-P. Kühn, and M. Spiliopoulou, "Learning and inspecting classification rules from longitudinal epidemiological data to identify predictive features on hepatic steatosis," *Expert Systems with Applications*, vol. 41, no. 11, pp. 5405–5415, 2014. (cited on Page 2)

[11] C.-F. Tsai and Y.-C. Hsiao, "Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches," *Decision Support Systems*, vol. 50, no. 1, pp. 258–269, 2010. (cited on Page 2)

[12] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in *Proceedings of the 17th International Conference on Machine Learning (ICML '00)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 359–366. (cited on Page 2, 5, 22, 34, 57, 64, 75, and 109)

[13] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016. (cited on Page 2)

[14] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Mining and*

Knowledge Discovery, vol. 28, no. 4, pp. 851–881, July 2014. (cited on Page 2 and 38)

[15] A. Bagnall and J. Lines, "An experimental evaluation of nearest neighbour time series classification," *arXiv preprint arXiv:1406.4757*, 2014. (cited on Page 2)

[16] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proceedings of the 23rd international conference on Machine learning.* ACM, 2006, pp. 1033–1040. (cited on Page 2)

[17] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Machine Learning Proceedings 1994.* Elsevier, 1994, pp. 121–129. (cited on Page 3 and 5)

[18] M. Berlingerio, F. Bonchi, F. Giannotti, and F. Turini, "Mining clinical data with a temporal dimension: a case study," in *IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2007).* IEEE, 2007, pp. 429–436. (cited on Page 3)

[19] I. Karlsson, P. Papapetrou, and H. Boström, "Forests of randomized shapelet trees," in *International Symposium on Statistical Learning and Data Sciences.* Springer, 2015, pp. 126–136. (cited on Page 3)

[20] F. Höppner, "Time series abstraction methods-a survey." in *GI Jahrestagung*, 2002, pp. 777–786. (cited on Page 3, 6, and 37)

[21] A. K. Tung, J. Han, L. V. Lakshmanan, and R. T. Ng, "Constraint-based clustering in large databases," in *Proceedings of the 8th International Conference on Database Theory (ICDT 2001).* Springer, 2001, pp. 405–419. (cited on Page 4 and 41)

[22] D. Koller and M. Sahami, "Toward optimal feature selection," in *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning (ICML'96).* Morgan Kaufmann Publishers Inc., 1996, pp. 284–292. (cited on Page 5)

[23] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005. (cited on Page 5)

[24] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16 – 28, 2014, 40th-year commemorative issue. (cited on Page 5 and 90)

[25] H. H. Inbarani, A. T. Azar, and G. Jothi, "Supervised hybrid feature selection based on pso and rough sets for medical diagnosis," *Computer methods and programs in biomedicine*, vol. 113, no. 1, pp. 175–185, 2014. (cited on Page 5)

[26] C. C. Aggarwal and J. Han, *Frequent Pattern Mining.* Springer Publishing Company, Incorporated, 2014. (cited on Page 6)

[27] T. Hielscher, M. Spiliopoulou, H. Völzke, and J.-P. Kühn, "Mining Longitudinal Epidemiological Data to Understand a Reversible Disorder," in *Proceedings of the 13th International Symposium on Intelligent Data Analysis (IDA2014).* Springer, 2014, pp. 120–130. (cited on Page 11, 12, 13, 46, 74, and 106)

[28] T. Hielscher, H. Völzke, P. Papapetrou, and M. Spiliopoulou, "Discovering, selecting and exploiting feature sequence records of study participants for the classification of epidemiological data on hepatic steatosis," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing (SAC '18).* ACM, 2018, pp. 6–13. (cited on Page 11, 12, 49, and 56)

[29] U. Niemann, T. Hielscher, M. Spiliopoulou, H. Völzke, and J. P. Kühn, "Can we classify the participants of a longitudinal epidemiological study from their previous evolution?" in *Proceedings of the 28th IEEE Symposium on Computer-Based Medical Systems (CBMS2015).* IEEE, 2015, pp. 121–126. (cited on Page 11, 12, 13, 46, 50, and 106)

[30] T. Hielscher, M. Spiliopoulou, H. Völzke, and J.-P. Kühn, "Identifying relevant features for a multi-factorial disorder with constraint-based

subspace clustering," in *Proceedings of the 29th IEEE Symposium on Computer-Based Medical Systems (CBMS2016)*. IEEE, 2016, pp. 207–212. (cited on Page 12, 13, and 74)

[31] T. Hielscher, U. Niemann, B. Preim, H. Völzke, T. Ittermann, and M. Spiliopoulou, "A framework for expert-driven subpopulation discovery and evaluation using subspace clustering for epidemiological data," *Expert Systems with Applications*, vol. 113, pp. 147 – 160, 2018. (cited on Page 12, 15, and 98)

[32] S. Alemzadeh, T. Hielscher, U. Niemann, L. Cibulski, T. Ittermann, H. Völzke, M. Spiliopoulou, and B. Preim, "Subpopulation discovery and validation in epidemiological data." in *Proceedings of the 19th EG/VGTC Conference on Visualization (EuroVis)*. Eurographics Association, 2017, pp. 43–47. (cited on Page 13 and 74)

[33] D. R. Wilson and T. R. Martinez, "Improved heterogeneous distance functions," *Journal of Artificial Intelligenece Research*, vol. 6, no. 1, pp. 1–34, Jan. 1997. (cited on Page 19)

[34] P.-N. Tan, V. Kumar, and M. Steinbach, *Introduction to data mining*, 1st ed. Boston : Pearson Addison Wesley, 2005. [Online]. Available: https://books.google.de/books?id=Wx4NPK4qHXsC (cited on Page 28)

[35] D. Tomar and S. Agarwal, "A survey on data mining approaches for healthcare," *International Journal of Bio-Science and Bio-Technology*, vol. 5, no. 5, pp. 241–266, 2013. (cited on Page 28)

[36] M. Sasaki and H. Shinnou, "Spam detection using text clustering," in *Proceedings of the 2005 International Conference on Cyberworlds (CW '05)*. IEEE, 2005, pp. 316–319. (cited on Page 28)

[37] V. Hanagandi, A. Dhar, and K. Buescher, "Density-based clustering and radial basis function modeling to generate credit card fraud scores," in *Proceedings of the 1996 Conference on Computational Intelligence for Financial Engineering (IEEE/IAFE 1996)*. IEEE, 1996, pp. 247–251. (cited on Page 28)

[38] G. W. Milligan and M. C. Cooper, “Methodology review: Clustering methods,” *Applied psychological measurement*, vol. 11, no. 4, pp. 329–354, 1987. (cited on Page 28)

[39] R. Xu and D. Wunsch, “Survey of clustering algorithms,” *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005. (cited on Page 28)

[40] P. Berkhin *et al.*, “A survey of clustering data mining techniques.” *Grouping multidimensional data*, vol. 25, p. 71, 2006. (cited on Page 28, 29, and 31)

[41] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: a review,” *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999. (cited on Page 28)

[42] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988. (cited on Page 29)

[43] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A k-means clustering algorithm,” *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979. (cited on Page 29, 41, and 58)

[44] H.-S. Park and C.-H. Jun, “A simple and fast algorithm for k-medoids clustering,” *Expert systems with applications*, vol. 36, no. 2, pp. 3336–3341, 2009. (cited on Page 29)

[45] M. Norouzi and D. J. Fleet, “Cartesian k-means,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR ’13)*. Portland, OR, USA: IEEE, 2013, pp. 3017–3024. (cited on Page 29)

[46] D. Yu, G. Liu, M. Guo, and X. Liu, “An improved k-medoids algorithm based on step increasing and optimizing medoids,” *Expert Systems with Applications*, vol. 92, no. Supplement C, pp. 464–473, 2017. (cited on Page 29)

[47] A. Hinneburg and D. A. Keim, “An efficient approach to clustering in large multimedia databases with noise,” in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD’98).* AAAI Press, 1998, pp. 58–65. (cited on Page 29)

[48] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *KDD’96: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining.* Portland, Oregon, USA: AAAI Press, 1996, pp. 226–231. (cited on Page 29, 30, 49, 50, and 81)

[49] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, “Optics: Ordering points to identify the clustering structure,” in *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD ’99).* Philadelphia, PA, USA: ACM, 1999, pp. 49–60. (cited on Page 29)

[50] S. T. Mai, I. Assent, and M. Storgaard, “Anydbc: an efficient anytime density-based clustering algorithm for very large complex datasets,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’16).* ACM, 2016, pp. 1025–1034. (cited on Page 29)

[51] E. Schikuta and M. Erhart, “The bang-clustering system: Grid-based data analysis,” in *Proceedings of the 2nd International Symposium on Intelligent Data Analysis (IDA 97)*, X. Liu, P. Cohen, and M. Berthold, Eds. London, UK: Springer Berlin Heidelberg, 1997, pp. 513–524. (cited on Page 29)

[52] A. Pilevar and M. Sukumar, “Gchl: A grid-clustering algorithm for high-dimensional very large spatial data bases,” *Pattern Recognition Letters*, vol. 26, no. 7, pp. 999 – 1010, 2005. (cited on Page 30)

[53] L. Parsons, E. Haque, and H. Liu, “Subspace clustering for high dimensional data: A review,” *SIGKDD Exploration Newsletter*, vol. 6, no. 1, pp. 90–105, Jun. 2004. (cited on Page 31 and 33)

[54] H.-P. Kriegel, P. Kröger, and A. Zimek, “Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and

correlation clustering," *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 1, pp. 1:1–1:58, 2009. (cited on Page 32 and 33)

[55] K. Kailing, H.-P. Kriegel, P. Kroeger, and S. Wanka, "Ranking interesting subspaces for clustering high dimensional data," in *Proceedings of the 7th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2003)*. Springer, 2003, pp. 241–252. (cited on Page 33)

[56] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *SIGMOD'98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*. Seattle, WA, USA: ACM, 1998, pp. 94–105. (cited on Page 33)

[57] S. Goil, H. Nagesh, and A. Choudhary, "Mafia: Efficient and scalable subspace clustering for very large data sets," in *KDD'99: Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*. San Diego, CA, USA: ACM, 1999, pp. 443–452. (cited on Page 33)

[58] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," *SIGMOD Rec.*, vol. 28, no. 2, pp. 61–72, 1999. (cited on Page 33)

[59] K. gu Woo, J. hoon Lee, M. ho Kim, and Y. joon Lee, "Findit: a fast and intelligent subspace clustering algorithm using dimension voting," in *PhD thesis, Korea Advanced Institute of Science and Technology*, 2002. (cited on Page 33)

[60] J. Schafer, "The application of data-mining to recommender systems," *Encyclopedia of data warehousing and mining*, vol. 1, pp. 44–48, 2009. (cited on Page 33)

[61] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002. (cited on Page 34)

[62] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997. (cited on Page 34)

[63] R. Abraham, J. B. Simha, and S. Iyengar, "Medical datamining with a new algorithm for feature selection and naïve bayesian classifier," in *Information Technology,(ICIT 2007). 10th International Conference on*. IEEE, 2007, pp. 44–49. (cited on Page 34)

[64] M. Zhang, Y. Yang, F. Shen, H. Zhang, and Y. Wang, "Multi-view feature selection and classification for alzheimer's disease diagnosis," *Multimedia Tools and Applications*, vol. 76, pp. 10 761–10 775, 2015. (cited on Page 34)

[65] R. Cai, Z. Zhang, and Z. Hao, "Bassum: A bayesian semi-supervised method for classification feature selection," *Pattern Recognition*, vol. 44, no. 4, pp. 811–820, 2011. (cited on Page 34)

[66] K. Sechidis and G. Brown, "Simple strategies for semi-supervised feature selection," *Machine Learning*, vol. 107, no. 2, pp. 357–395, 2018. (cited on Page 34)

[67] D. He, I. Rish, D. Haws, and L. Parida, "Mint: Mutual information based transductive feature selection for genetic trait prediction," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 13, no. 3, pp. 578–583, 2016. (cited on Page 35)

[68] M. Hindawi, K. Allab, and K. Benabdeslem, "Constraint selection-based semi-supervised feature selection," in *2011 IEEE 11th International Conference on Data Mining*. IEEE, Dec 2011, pp. 1080–1085. (cited on Page 35)

[69] M. Kalakech, P. Biela, L. Macaire, and D. Hamad, "Constraint scores for semi-supervised feature selection: A comparative study," *Pattern Recognition Letters*, vol. 32, no. 5, pp. 656 – 665, 2011. (cited on Page 90)

[70] K. Benabdeslem and M. Hindawi, "Efficient semi-supervised feature selection: Constraint, relevance, and redundancy," *IEEE Transactions*

on Knowledge and Data Engineering, vol. 26, no. 5, pp. 1131–1143, May 2014. (cited on Page 35)

[71] K. Orphanou, A. Stassopoulou, and E. Keravnou, "Temporal abstraction and temporal bayesian networks in clinical domains: A survey," *Artificial Intelligence in Medicine*, vol. 60, no. 3, pp. 133–149, March 2014. (cited on Page 38)

[72] A. Singh, G. Nadkarni, O. Gottesman, S. B. Ellis, E. P. Bottinger, and J. V. Guttag, "Incorporating temporal ehr data in predictive models for risk stratification of renal function deterioration," *J. of Biomedical Informatics*, vol. 53, pp. 220–228, November 2015. (cited on Page 39)

[73] J. Zhao, P. Papapetrou, L. Asker, and H. Boström, "Learning from heterogeneous temporal data in electronic health records," *J. of Biomedical Informatics*, vol. 65, pp. 105–119, January 2017. (cited on Page 38)

[74] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *DMKD '03: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.* ACM, June 2003, pp. 2–11. (cited on Page 38)

[75] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 107–144, October 2007. (cited on Page 38)

[76] I. Karlsson, P. Papapetrou, and H. Boström, "Generalized random shapelet forests," *Data Mining and Knowledge Discovery*, vol. 30, no. 5, pp. 1053–1085, Sep 2016. (cited on Page 38 and 39)

[77] R. Moskovitch and Y. Shahar, "Medical temporal-knowledge discovery via temporal abstraction," in *AMIA annual symposium proceedings*, vol. 2009. American Medical Informatics Association, 2009, p. 452. (cited on Page 38 and 119)

[78] I. Batal, L. Sacchi, R. Bellazzi, and M. Hauskrecht, "Multivariate time series classification with temporal abstractions." in *FLAIRS Conference*, 2009. (cited on Page 38)

[79] L. Ye and E. Keogh, "Time series shapelets: a new primitive for data mining," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2009, pp. 947–956. (cited on Page 39)

[80] S. Basu, I. Davidson, and K. Wagstaff, *Constrained clustering: Advances in algorithms, theory, and applications.* CRC Press, 2008. (cited on Page 41)

[81] J. Wu, S. Pan, X. Zhu, and Z. Cai, "Boosting for multi-graph classification," *IEEE transactions on cybernetics*, vol. 45, no. 3, pp. 416–429, 2015. (cited on Page 41)

[82] J. Sun, F. Wang, J. Hu, and S. Edabollahi, "Supervised patient similarity measure of heterogeneous patient records," *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 1, pp. 16–24, 2012. (cited on Page 41)

[83] C. Ruiz, M. Spiliopoulou, and E. Menasalvas, "C-DBSCAN: Density-Based Clustering with Constraints," in *Proceedings of the 11th International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing (RSFDGrC 2007).* Springer, 2007, pp. 216–22. (cited on Page 41 and 42)

[84] H. A. Dau, N. Begum, and E. Keogh, "Semi-supervision dramatically improves time series clustering under dynamic time warping," in *CIKM 2016: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management.* Indianapolis, IN, USA: ACM, 2016, pp. 999–1008. (cited on Page 42)

[85] J. Liu, C. E. Brodley, B. C. Healy, and T. Chitnis, "Removing confounding factors via constraint-based clustering: An application to finding homogeneous groups of multiple sclerosis patients," *Artificial Intelligence in Medicine*, vol. 65, no. 2, pp. 79–88, 2015. (cited on Page 42)

[86] A. Shama and S. Phadikar, "Automatic color image segmentation using spatial constraint based clustering," in *2014 Conference on Emerging Trends in Computing and Communication (ETCC 2014)*. Springer, 2014, pp. 113–121. (cited on Page 42)

[87] E. Fromont, A. Prado, and C. Robardet, "Constraint-based subspace clustering," in *SDM09: Proceedings of the 2009 SIAM International Conference on Data Mining*. Sparks, Nevada, USA: SIAM, 2009, pp. 26–37. (cited on Page 42 and 43)

[88] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl *et al.*, "Constrained k-means clustering with background knowledge," in *Proceedings of the 18th International Conference on Machine Learning (ICML '01)*, vol. 1. Morgan Kaufmann Publishers Inc., 2001, pp. 577–584. (cited on Page 43)

[89] S. Günnemann, I. Färber, M. Rüdiger, and T. Seidl, "SMVC: semi-supervised multi-view clustering in subspace projections," in *KDD '14: Proceedings of the 20th ACM SIGKDD conference on Knowledge discovery and data mining*. New York, New York, USA: ACM, 2014, pp. 253–262. (cited on Page 43)

[90] A. Struyf, M. Hubert, and P. Rousseeuw, "Clustering in an object-oriented environment," *Journal of Statistical Software, Articles*, vol. 1, no. 4, pp. 1–30, 1997. (cited on Page 59)

[91] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD 2000)*, ser. SIGMOD '00, New York, NY, USA, 2000, pp. 93–104. (cited on Page 59)

[92] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53 – 65, 1987. (cited on Page 59)

[93] H. Völzke, D. Alte, C. O. Schmidt, D. Radke, R. Lorbeer, N. Friedrich, N. Aumann, K. Lau, M. Piontek, G. Born *et al.*, "Cohort profile: the

study of health in pomerania," *International Journal of Epidemiology*, vol. 40, no. 2, pp. 294—-307, 2011. (cited on Page 60, 62, and 98)

[94] H. Völzke, "Multicausality in fatty liver disease: is there a rationale to distinguish between alcoholic and non-alcoholic origin," *World Journal of Gastroenterology*, vol. 18, no. 27, pp. 3492–501, 2012. (cited on Page 62)

[95] W. M. Project, "Who monica project: assessing chd mortality and morbidity," *International journal of epidemiology*, vol. 18, no. Supplement_1, pp. S38–S45, 1989. (cited on Page 62)

[96] H. Brønnum-Hansen, T. Jørgensen, M. Davidsen, M. Madsen, M. Osler, L. U. Gerdes, and M. Schroll, "Survival and cause of death after myocardial infarction: The Danish MONICA study," *Journal of clinical epidemiology*, vol. 54, no. 12, pp. 1244–1250, 2001. (cited on Page 62)

[97] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324 (cited on Page 64)

[98] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining.* ACM, 2016, pp. 785–794. (cited on Page 64)

[99] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995. (cited on Page 64)

[100] S. L. Salzberg, "C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993," *Machine Learning*, vol. 16, no. 3, pp. 235–240, Sep 1994. [Online]. Available: https://doi.org/10.1007/BF00993309 (cited on Page 64)

[101] "C5," http://rulequest.com, accessed: 2018-09-08. (cited on Page 64)

[102] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees.* Monterey, CA: Wadsworth and Brooks, 1984. (cited on Page 64)

[103] S. Haykin, *Neural networks: a comprehensive foundation.* Prentice Hall PTR, 1994. (cited on Page 64)

[104] H. Zhang, "The optimality of naïve bayes," in *Proceedings of the17th International Florida Artificial Intelligence Research Society Conference.* AAAI, 2004. (cited on Page 64 and 89)

[105] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992. [Online]. Available: http://www.jstor.org/stable/2685209 (cited on Page 64 and 89)

[106] E. Amid, A. Gionis, and A. Ukkonen, "A kernel-learning approach to semi-supervised clustering with relative distance comparisons," in *Proceedings of the 2015th European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2015).* Springer, 2015, pp. 219–234. (cited on Page 78)

[107] D. E. Ho, K. Imai, G. King, and E. A. Stuart, "Matchit: Nonparametric preprocessing for parametric causal inference," *Journal of Statistical Software*, vol. 42, no. 8, pp. 1–28, 2011. (cited on Page 86)

[108] R. Sheikhpour, M. A. Sarram, S. Gharaghani, and M. A. Z. Chahooki, "A survey on semi-supervised feature selection methods," *Pattern Recogn.*, vol. 64, no. C, pp. 141–158, Apr. 2017. [Online]. Available: https://doi.org/10.1016/j.patcog.2016.11.003 (cited on Page 89)

[109] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," in *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS'13).* Curran Associates Inc., December 2013, pp. 431–439. (cited on Page 90)

[110] M. Vanderpump, "The epidemiology of thyroid disease," *British Medical Bulletin*, vol. 99, no. 1, pp. 39–51, 2011. (cited on Page 99)

[111] R. Gutekunst, W. Becker, R. Hehrmann, T. Olbricht, and P. Pfannenstiel, “Ultraschalldiagnostik der schilddrüse,” *DMW-Deutsche Medizinische Wochenschrift*, vol. 113, no. 27, pp. 1109–1112, 1988. (cited on Page 99)

[112] M. Roden, “Mechanisms of disease: hepatic steatosis in type 2 diabetes—pathogenesis and clinical relevance,” *Nature Reviews Endocrinology*, vol. 2, no. 6, pp. 335–348, 2006. (cited on Page 106)

[113] H. Völzke, S. Schwarz, S. E. Baumeister, and et al., “Menopausal status and hepatic steatosis in a general female population,” *Gut*, vol. 56, no. 4, pp. 594–595, 2007. (cited on Page 106)

[114] P. J. Meffert, S. E. Baumeister, M. M. Lerch, J. Mayerle, W. Kratzer, and H. Völzke, “Development, external validation, and comparative assessment of a new diagnostic score for hepatic steatosis,” *The American Journal of Gastroenterology*, vol. 109, no. 9, pp. 1404–1414, 2014. (cited on Page 106)

[115] L. Zheng, W. Yan, Y. Kong, P. Liang, and Y. Mu, “An epidemiological study of risk factors of thyroid nodule and goiter in chinese women,” *International journal of environmental research and public health*, vol. 12, no. 9, pp. 11 608–11 620, 2015. (cited on Page 109)

[116] D. Rendina, G. De Filippo, G. Mossetti, and et al., “Relationship between metabolic syndrome and multinodular non-toxic goiter in an inpatient population from a geographic area with moderate iodine deficiency,” *Journal of Endocrinological Investigation*, vol. 35, no. 4, pp. 407–412, 2012. (cited on Page 109)

[117] R. Moskovitch, C. Walsh, G. Hripcsak, and N. Tatonetti, “Prediction of biomedical events via time intervals mining,” in *ACM KDD Workshop on Connected Health in Big Data Era*, 2014. (cited on Page 119)

[118] X. He, D. Cai, and P. Niyogi, “Laplacian score for feature selection,” in *Proceedings of the 18th International Conference on Neural Information Processing Systems (NIPS’05)*. ACM, 2005, pp. 507–514. (cited on Page 120)

Ehrenerklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; verwendete fremde und eigene Quellen sind als solche kenntlich gemacht. Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.
Ich habe insbesondere nicht wissentlich:

- Ergebnisse erfunden oder widersprüchliche Ergebnisse verschwiegen,
- statistische Verfahren absichtlich missbraucht, um Daten in ungerechtfertigter Weise zu interpretieren,
- fremde Ergebnisse oder Veröffentlichungen plagiiert,
- fremde Forschungsergebnisse verzerrt wiedergegeben.

Mir ist bekannt, dass Verstöße gegen das Urheberrecht Unterlassungs- und Schadensersatzansprüche des Urhebers sowie eine strafrechtliche Ahndung durch die Strafverfolgungsbehörden begründen kann. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Magdeburg, den 26.03.2019

Tommy Hielscher