David Cemernek, Mag.

Outlier Detection as Instance Selection Method for Feature Selection in Time Series Classification

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Dr. Roman Kern

Institute of Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, February 2019

This document is set in Palatino, compiled with pdfLaTeX2e and `Biber`.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Abstract

In order to allow machine learning algorithms to extract knowledge from raw data, these data must first be cleaned, transformed, and put into machine-appropriate form. These often very time-consuming phase is referred to as preprocessing. An important step in the preprocessing phase is feature selection which aims at better performance of prediction models by reducing the amount of features of a data set. Within these datasets instances of different events are often imbalanced, which means that certain normal events are over-represented while other rare events are very limited. Typically these rare events are of special interest since they have more discriminative power than normal events. The aim of this work was to filter instances provided to feature selection methods for these rare instances and thus positively influence the feature selection process. In the course of this work, we were able to show that this filtering has a positive effect on the performance of classification models and that outlier detection methods are suitable for this filtering. For some data sets, the resulting increase of performance was only a few percent, but for other datasets, we were able to achieve increases in performance of up to 16 percent. This work should lead to the improvement of the predictive models and the better interpretability of feature selection in the course of the preprocessing phase. In the spirit of open science and to increase transparency within our research field, we have made all our source code and the results of our experiments available in a publicly available repository.

Zusammenfassung

Damit Machine Learning Algorithmen Wissen aus Rohdaten extrahieren können, müssen diese Daten zunächst bereinigt, transformiert und in maschinengerechte Form gebracht werden. Diese oft sehr zeitaufwändige Phase wird als Preprocessing bezeichnet. Ein wichtiger Schritt in der Preprocessing-Phase ist Feature Selection, die auf eine bessere Leistung von Vorhersagemodellen abzielt, indem die Anzahl der Features eines Datensatzes reduziert wird. In diesen Datensätzen sind die Instanzen verschiedener Ereignisse oft unausgewogen, was bedeutet, dass bestimmte normale Ereignisse überrepräsentiert sind, während andere seltene Ereignisse sehr begrenzt vorkommen. Typischerweise sind diese seltenen Ereignisse von besonderem Interesse, da sie einen höheren Informationsgehalt besitzen als normale Ereignisse. Ziel dieser Arbeit ist es die vorhandenen Instanzen auf diese seltenen Instanzen zu filtern, um dadurch die Feature Selection positiv zu beeinflußen. Im Verlauf dieser Arbeit konnten wir zeigen, dass sich diese Filterung positiv auf die Leistungsfähigkeit von Klassifikationsmodellen auswirkt und dass Ausreißermittlungsverfahren für diese Filterung geeignet sind. Bei einigen Datensätzen betrug die Leistungssteigerung nur wenige Prozent, bei anderen Datensätzen konnten wir jedoch eine Leistungssteigerung von bis zu 16 Prozent erzielen. Diese Arbeit sollte zur Verbesserung der Vorhersagemodelle und zur besseren Interpretierbarkeit der Feature Selection im Verlauf des Preprocessings führen. Im Sinne einer offenen Wissenschaft und zur Erhöhung der Transparenz in unserem Forschungsbereich haben wir unseren gesamten Quellcode und die Ergebnisse unserer Experimente in einem öffentlich zugänglichen Repository verfügbar gemacht.

Contents

List of Figures

List of Figures

1. Introduction

1.1. Problem and Motivation

In order to gain important insights from data, it is not enough to simply feed appropriate algorithms with raw data. Rather extensive and expensive data cleansing, data format preparation, and data transformation are required for these algorithms to work effectively. The phase in which all this tedious work is carried out is referred to as preprocessing.
One of the most important steps in preprocessing is the selection of variables to be used for later classification or regression modeling. One way to select these variables is to have them manually filtered by domain experts.
In our research unit we are mostly dealing with industrial damage detection, for which the data is represented as sensor data recorded over time referred to as time series data [CBK09]. An important characteristic of time series data is its high dimensionality, which is a reason why manual filtering of variables by domain experts does not scale. Hence manual filtering is too slow, too expensive, prone to errors, and scales only up to a certain number of variables [14].
The task for selecting the most relevant variables or features of a dataset is referred to as feature selection. The feature space is therefore divided into relevant and irrelevant or redundant features. Irrelevant or redundant features do not provide additional information and can be left out during modeling of a problem [SG16]. Even worse, irrelevant or redundant features could lead to incorrect predictions and hence have a negative impact on the prediction performance of models [AAB11].
To select the most relevant features, feature selection depends on the provided instances. However in our data-driven world, where data and available variables are constantly growing, we are able to provide millions or even hundreds of millions of data instances to feature selection

algorithms. The vast majority of these instances are represented by normal and non-discriminatory data, while rare and discriminatory cases only account for a small proportion [14].
Many algorithms in machine learning, including feature selection algorithms, are not able to handle these rare cases since they are not adequately represented [Wei04]. With instance selection there already exists a research field which has the main goal to select only these rare and most discriminative instances. However, since instance selection methods are typically used to compile training sets for classification or regression models, they have a strong dependency on these models and are not designed for the task of feature selection.
A quite similar research field to instance selection is outlier detection. The main goal of outlier detection is to find observations which deviate so much from other observations as to arouse suspicious that it was generated by a different mechanism [ANH16].
In order to circumvent the dependency of instance selection methods to models, we apply outlier detection methods to select the most discriminative instances for feature selection. The idea behind our approach is that outlier detection should rank the instances for their discriminative power and should provide only a specific percentage of these ranked instances to the feature selection algorithms. These feature selection algorithms select the relevant features, which are then provided to classification models. The results of these models are then compared to the results of "conventional" feature selection and classification models.

1.2. Research Questions

The previously presented problem can be summarized in the form of our main research question as follows: **Is it possible to positively increase the performance of learning algorithms by providing only a specific subset of our training instances to feature selection algorithms?** This main research question has two aspects:

- Are outlier detection algorithms suitable for instance selection to influence feature selection algorithms?

- Do these filtered feature subsets have a positive impact on the performance of learning algorithms?

1.3. Structure of Work

The rest of this work is organized as follows: In order to create the foundation for our approach, we first of all need a basic understanding of the involved topics. Therefore we give an overview of time series analysis, feature selection, instance selection and outlier detection in Chapter 2. Then we introduce the method for our state of the art analysis concerning instance selection based on outlier detection methods. Finally this chapter closes with the presentation of results of our state of the art analysis. We then give a detailed insight into the method we have developed in Chapter 3. In the following chapter 4 we describe in detail the evaluation and the experimental setup to evaluate our developed approach. This work will then be completed by a thorough summary and a outlook for future work in Chapter 5.

2. Related Work

2.1. Methodology of Literature Review

In order to create the expose for this work a simplified literature search was carried out. In this literature research, the main goal was to get an overview of the relevant topics, problems and synonyms. This literature research was carried out in free and non-systematic way using the following websites/tools:

- google scholar [1]
- wikipedia [2]
- open knowledge maps [3]

Based on the obtained results, the respective areas for each of which a separate extended literature research had to be performed, were broken down. As the name of the work suggests, there are three major topics: time series analysis, feature selection, and outlier detection. Since the selection of training instances is a separate research area, namely instance selection, this in total adds up to four main research topics. The systematic approach was identical for all four major topics. Based on the short literature search we collected the synonyms for each topic. As example we provide the synonyms for time series analysis:

- Time-series data or Time series data
- Time-series prediction or Time series prediction
- Time-series modeling or Time series modeling
- Time-series data modeling or Time series data modeling

[1] https://scholar.google.at
[2] http://en.wikipedia.org
[3] https://openknowledgemaps.org

- Time-series data mining or Time series data mining

Based on the obtained synonyms a search matrix (= combination of the different synonyms) was developed. The entries of the search matrix correspond to the search terms that were used with different parameters in the various platforms. The presented platforms are based on the research on publications of the authors within the computer science and machine learning field:

- ScienceDirect [4]
- arXiv [5]
- DE Gruyter [6]
- Scopus [7]
- IEEE Xplore [8]
- EmeraldInstight [9]
- SpringerLink [10]
- Google Scholar [11]
- Directory of Open Access Journals [12]
- Web of Science [13]

For each topic the search terms from the search matrix were combined with general search parameters and platform specific parameters. The two most used general search parameters were the year (from 2018-2008) and the field where the search term should be searched for (mostly we searched in the title and abstract fields). Specific search parameters often included the subject to search in, for example computer science, or the document type for example, article, review or survey. In order to better understand the literature search, we have listed an example search with results in Section A.5 in the Appendix. For each topic, the number of matches found

[4] https://www.sciencedirect.com/search/advanced
[5] https://arxiv.org/search/advanced
[6] https://www.degruyter.com/dg/advancedsearchpage
[7] https://www.scopus.com/search/form.uri?display=basic
[8] https://ieeexplore.ieee.org/search/advsearch.jsp
[9] https://www.emeraldinsight.com/search/advanced
[10] https://link.springer.com/advanced-search
[11] https://scholar.google.at
[12] https://doaj.org/search
[13] http://login.webofknowledge.com (needs account)

per platform, the number of relevant results and the titles of the relevant results were documented in an overview table. The relevant results were further specified in the "relevant table" (for example, type of article, year of publication, filename, abstract,...). Based on the more detailed specification of the relevant results and the abstract, it was determined whether the paper was relevant for our work or not. In order to get the best possible overview of the topic, the focus was on surveys, evaluations, and reviews. The relevant publications were systematically studied including also their references. Together with recommendations from the used platforms [14] and our colleagues these references formed two very valuable additional cross-platform sources. Especially the recommendations of the used search platforms led to some promising work. For the overview given outlier detection we already had a very broad collection from previous research. These collection contained already 21 surveys, reviews, evaluations, and articles, which was clearly sufficient for a purpose.

2.2. Background

2.2.1. Time Series Analysis

Time series analysis is a branch of statistics which deals with the analysis of **time series data** [FV17]. Before we elaborate on the numerous areas and techniques of time series analysis, we first discuss the basic characteristics of time series data. Time series data has a natural (temporal) ordering. This ordering between different observations basically distinguishes time series data from non-time serial data [Dag10]. For time series data, the dimension of time is explicitly taken into account, respectively given the definition according to [Fu11] time series data:

- are collections of chronological observations
- are considering data as a whole instead of collection of numerical fields
- are ordered over time

[14] Recommendations from platforms such as researchgate https://www.researchgate.net/ or mendeley https://www.mendeley.com

- consider time as the primary axis

We provided an artificially generated example of time series data in the plot shown in Figure 2.1.

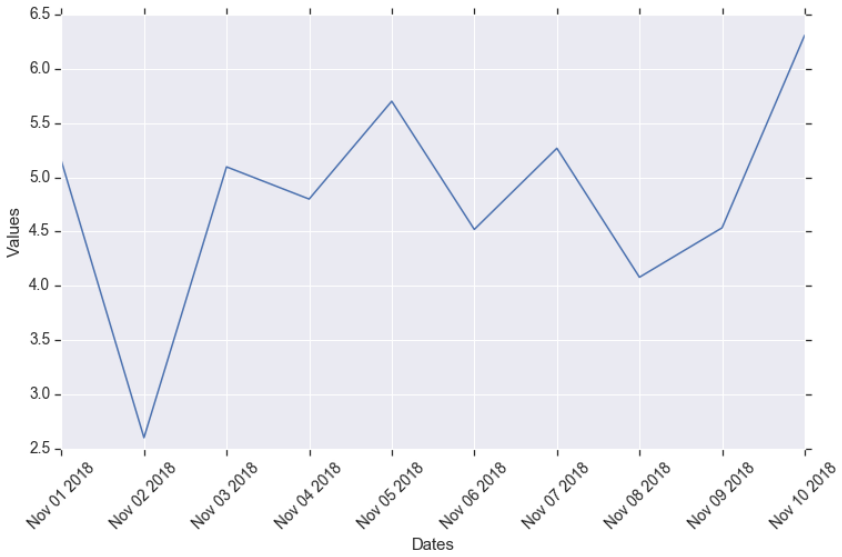


Figure 2.1.: Figure shows some randomly generated time series data. The x-axis represents the time dimension with days as interval. The y-axis represents the artificially generated values of the different observations.

Given that a large percentage of the data produced worldwide is time series data and the exponentially growing size of databases, there has recently been an explosion of interest in time series analysis. Among many others the following data from various domains are examples of time series data [Rat+09]:

- Finance: Presentation of the development of the stock market price of a company over time.
- Meteorology: Temperature development over time for a specific area like a country, state, or city.
- Trade: Historical store sales data, for example sold products over time.
- Medical: Electrocardiograms showing the electrical activity of a heart over time.

Referring to [Fu11] there are various related research areas concerning time series data, namely finding similar time series, sub-sequence searching in time series, dimensionality reduction, and segmentation. Generally speaking processing of (time series) data is only expedient if it is processed for the extraction of information and subsequently for the discovery of knowledge. The extraction of information in datasets is referred to as **data mining** [FPS96]. The equivalent for time series data is consequently denoted **time series data mining**. [EA12] specify the purpose of time series data mining as to extract meaningful knowledge from the *shape* of time series data. Following the definition of [EA12] time series data mining involves the following major tasks:

- Classification
- Clustering
- Motif discovery
- Outlier or anomaly detection
- Prediction
- Query by content
- Segmentation

Both [Fu11] and [FV17] also assign the following topics to the major time series data mining related tasks:

- Rule discovery
- Summarization

In [Fu11] the authors define pattern discovery the most common mining task, with clustering as the most common method for pattern discovery. Furthermore the authors subsume the tasks anomaly detection, motif discovery and finding discords under the term "pattern discovery".
Within the following paragraphs we will provide a brief description for each task related to time series data mining. **Classification** is the most popular data mining technique. Due to the fact that the classes are determined in advance, classification is also referred to as supervised learning. A classification algorithm assigns these predefined classes represented by so called labels to data instances or within time series domain to time series points or sub-sequences [Rat+09]. Since classification of time series data is one of the focus topics of this work, we will go into more detail in section 2.2.1.

Clustering algorithms find natural groups, called clusters in data. In contrary to classification there are no predefined labels to mark classes in data, thus clustering is also referred to as unsupervised learning. Clustering aims to organize instances within a cluster homogeneously (similar instances should belong to the same cluster), but the clusters should be as heterogeneous as possible (different clusters should be as distinct as possible) [EA12].
Motif discovery deals with finding recurring patterns in subsequences of time series data. This recurring patterns are referred to as "motifs" [FV17].
Outlier detection focuses on finding abnormal (or anomalous) sequences in time series data. A first step in anomaly detection is often to create a model for detecting normal time series and then find subsequences which deviates from this normal behavior [FV17]. We have a closer look at this topic in Sub-Section 2.2.4.
Prediction of subsequent or future values of time series data is based on the principle that observations close together in time are more closely related than observations far away from each other [Dag10]. Prediction tasks are modeling these correlations and dependencies between time series data in order to forecast future values [FV17]. Thus this task is also referred to as **time series forecasting**.
Query by content deals with finding of similar time series or time series sub-sequences given a query time series. To define similar time series query by content depends on the definition of a similarity measures between time series data [EA12].
Rule discovery is also referred to as association rule mining, and aims at finding relations between variables in (time series) data. [FV17]
Segmentation focus on creating approximations of time series data, by means of dimensionality reduction of potential high-dimensional time series data. [EA12]. Within the work of [Rat+09] segmentation is also referred to as **summarization**.

One fundamental problem for almost all of the above tasks of time series data mining is the representation of time series data. A common approach thereby is to transform the time series via some sort of dimensionality reduction followed by various indexing mechanisms. These techniques are complemented by the areas of similarity measures between time series and segmentation of time series data. [Fu11]. These aforementioned steps almost match the definition of [EA12], in which the three major issues for dealing

with (high-dimensional) time series data are:

- Data representation: Representation or reduction of high-dimensionality data to less dimensions, without changing the basic shape characteristics of the original time series data, for example via sampling or linear regression.
- Indexing methods: Organize massive sets of time series data for fast querying, for example via minimum bounding rectangle.
- Similarity measurements: Distinguish or match different pairs of time series data, for example via euclidean distance or dynamic time warping [15].

The same authors ([EA12]) denote these major issues as "implementation components" which represent the core aspects of time series data systems. Recapitulating [Fu11] the following two components can also be considered important components concerning time series data:

- Segmentation[16]: Also referred to as **summarization**, which performs helpful and necessary discretization of time series data. These techniques span from trivial summarization, such as calculating of summary statistics (for example the mean or variance of windows of time series data) to more sophisticated methods using natural language for summarizing time series data [Rat+09]
- Visualization: Presents time series data for further analysis to human users, for example via cluster based visualizations.

[FV17] denotes representation, similarity measures and the accompanying data mining tasks the three main research orientations in time series data mining. For a more in-depth look at the underlying components and tasks of time series data mining we can highly recommend the already cited papers: [Rat+09], [Fu11], [EA12], and [FV17].

[15] Dynamic time warping is an algorithm for mapping sequences of different lengths onto each other, for details see reference [BL14]

[16] According to [Fu11] segmentation can both be considered as a trend analysis technique and as a preprocessing step for some data mining tasks, which qualifies it also as an implementation component.

Time Series Classification

Time series classification (TSC) differs from traditional classification in that the elements to be classified are ordered. This ordering may be used for discriminant features [Bag+17]. In "traditional" classification this ordering of features is not important, and furthermore interaction between features is considered independent of their relative positions [BL14].
A possibly more intuitive comparison of traditional classification to TSC is based on the assumption that the former only uses static features, whereas TSC uses dynamic features, for which the change in values over time is relevant.
The three main characteristics that make TSC so difficult are: the small number of cases, large number of features and the highly correlated and/or redundant features. In traditional classification we already have good solutions for these three characteristics. Traditional classification algorithms typical have problems with discriminating features in autocorrelation, phase independence in classes, and embedded discriminative sub-series. Nevertheless, this does not mean that these problems are present in every time series dataset. This qualifies traditional classification algorithms as valuable baseline approaches, and these algorithms may provide important insight into problem characteristics of a specific dataset [Bag+17].
Until recently the default baseline algorithm for TSC was the 1-Nearest Neighbor classifier (1-NN) with euclidean distance. The **Nearest Neighbor Classifier** is a representative of instance-based learning algorithms. Algorithms of this kind only store training instances during their training phase. To classify a new (unseen) instance this new instance is compared with its closest neighbors within the stored training instances. To compare the closeness to given neighbors instance-based learners are using various different distance functions (probably the most famous one is the euclidean distance function) [BM02].
Since the authors of [BL14] stated that 1-NN classifier is easy to beat it should not be used as a baseline for TSC any more. Instead the authors recommended the usage of **1-Nearest Neighbor with dynamic time warping window (DTW1NN)** set through cross-validation as a more meaningful baseline. Furthermore due to the solid results, the authors selected **Rotation Forest** as their second benchmark algorithm. Rotation Forest is a variant of

ensemble learning. Ensemble learners use several (different) base learners, hence the term "ensemble". Within Rotation Forest the base learners are Decision Trees. For each base learner the feature set is randomly split and Principal Component Analysis is performed on each subset. This transformation forms new features for each base learner, since different feature sets will lead to different transformed features and thus different Decision Trees. [RKA06]

2.2.2. Feature Selection

Feature selection is the process of selecting the most relevant features from a dataset. More specific, feature selection should also remove irrelevant and redundant features [Guy+06].
Based on [GE03], we use the following terms: **Variable** refers to the raw input variable and **feature** refers to the some-how preprocessed or transformed variables. Next to variable the term "**attribute**" is used as a synonym for feature. In consequence to these concepts the following terms for feature selection can be found in literature: variable selection, attribute selection and variable or attribute subset selection.
Following [Guy+06] feature selection next to **feature construction** define the overall concept of **feature extraction**. Feature construction creates the representation of the data to model, for example defines transformations like standardization, normalization, and discretization. Another term often used in the environment of feature extraction is "**feature engineering**". Feature engineering is the process of applying domain knowledge to a problem to obtain the best representation of features that are used by models.
Since feature selection is a main part of this work, we only delve into the main concepts for it. The main objectives of feature selection are [GE03]:

- improving the prediction performance of predictors
- providing faster and more cost-effective predictors
- providing a better understanding of the underlying process that generated the data.

These objectives result in manifold benefits, such as [GE03]:

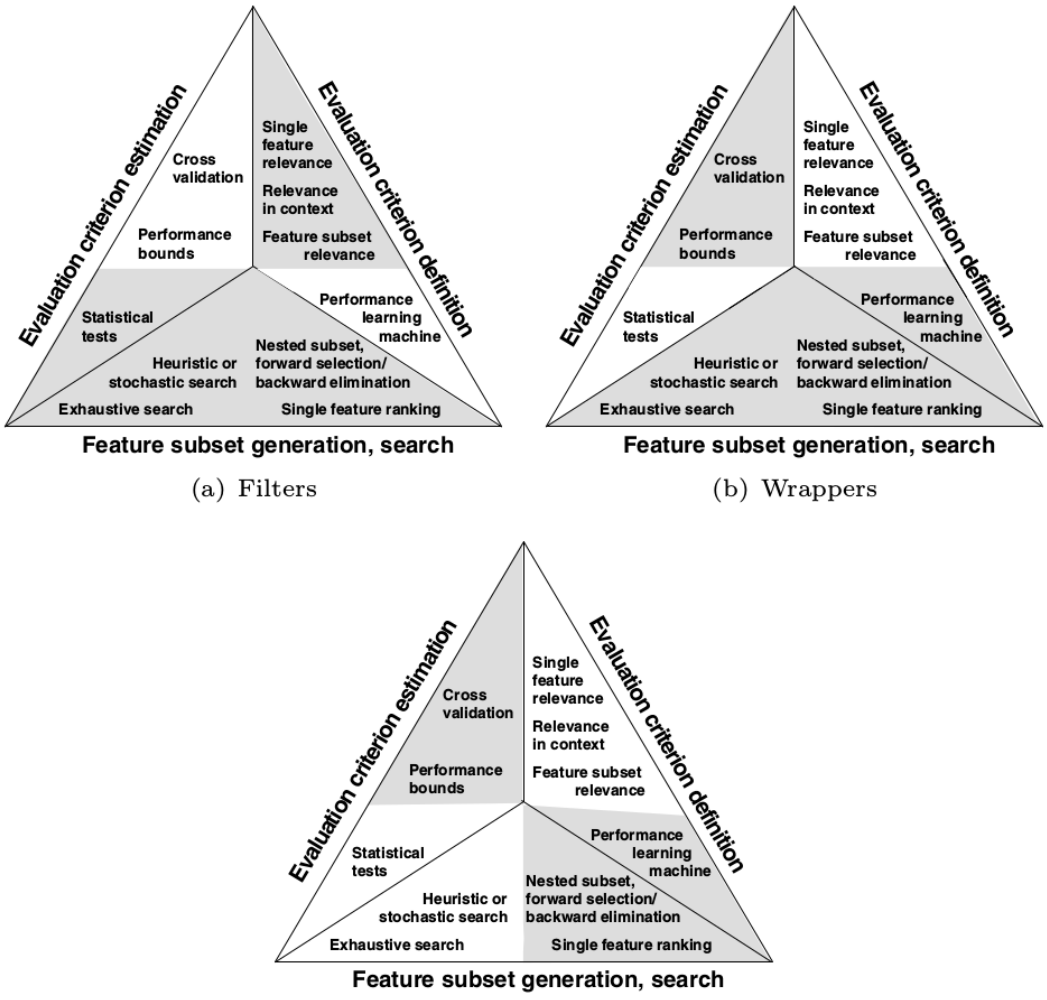
- facilitating data visualization and data understanding

- reducing the measurement and storage requirements
- reducing training and utilization times
- defying the curse of dimensionality to improve prediction performance
- and allow simpler, less complex models.

It is worth mentioning that some predictors may turn inefficient or even inapplicable in terms of memory and time if the number of features is too large. Even worse, irrelevant features could confuse some predictors, leading to incorrect predictions [AAB11].

Feature Selection Types

The three main types of feature selection, namely filter, wrapper and embedded methods are presented an overview in Figure 2.2.
Filter methods select features without optimizing the performance of a predictor. Mostly filter methods are applied in the form of feature ranking methods which rank features individually (univariate) and take into account their relation to a given target value in regression or labels in classification scenarios [GE03]. Examples for feature ranking methods are correlation ranking (for example Pearson or Spearman), Information theoretic ranking criteria (for example Mutual Information) or single feature classifiers ($R(i)^2$ ranking criteria). These methods are complemented by multivariate filter methods like the "Relief" algorithm [Guy+06].
Since filter methods only have to calculate m number of ranks, where m represents the number of features in a dataset, these methods are considered fast and effective, especially when m is large and the corresponding number of training examples is rather small (for example thousands of features and only hundreds of examples) [Guy+06]. Additionally filter methods do not depend on specific learning algorithms and therefore avoid over-fitting to given training data [VE14].
One drawback of filter methods is the potential risk that the selected subset is not optimal and that redundant features are selected. Features that are not relevant on their own may be relevant in combination with other features. This could lead to sub-optimal feature sets, but these subsets may be good enough in many cases [Guy+06].



(a) Filters

(b) Wrappers

(c) Embedded methods

Figure 2.2.: The tree principal types of feature selection. Gray shading shows the components which the three approaches use. Graphics extracted from [Guy+06]

Wrapper methods are using learning algorithms for the evaluation of feature subsets. The learning algorithm is used as a black-box, for example the evaluation is based on the classification rate of a classification algorithm on a defined test set. This evaluation is performed for each subset which may result in high computational costs and depends on the used learning algorithm especially for high-dimensional datasets [VE14].
Filter and wrapper methods make use of search strategies to search through the feature space. This search through all potential 2^N subsets for given data represents a NP-hard problem. This makes the evaluation of all subset inefficient or almost impossible. Often filter methods are "limited" to feature ranking methods for which the subsets only consist of currently rated feature, although hybrid methods exist, in which filters are used to create feature subsets [Guy+06].
In contrast to filter and wrapper methods, **embedded methods** do not separate the learning/training phase of a learning algorithm from the feature selection phase (for example Decision Trees). Embedded methods are less expensive in terms of computational requirements than wrapper methods, but still much slower than filter methods and the selected features strongly depend on the involved learning algorithm [VE14].
Whereas feature selection reduces the features of a given dataset, the main objective of **instance selection** is the reduction of instances, which is described in the next sub-section.

2.2.3. Instance Selection

In classification we need training instances to train a given model to create knowledge which is used to classify new instances. Some of these training instances do not increase or even worse negatively affect our knowledge and therefore are not useful for classification models. The process of selecting only relevant, removing or ignoring useless training instances is known as **instance selection** [Olv+10].
Although instance selection and feature selection are independent of each other they are often applied jointly to increase the dimensions of data. Instance selection is a subfield in the research area of data reduction. Data sampling for example is considered a data reduction technique but is not

an instance selection technique, since sampling randomly reduces data, whereas instance selection involves an intelligent process of categorization of instances, according to a degree of irrelevance or noise [GLH15].
Similar to feature selection methods, instance selection methods can be divided regarding the underlying method used for evaluation of instances [Olv+10]:

- Filter: Selection criterion uses a selection function independent of a classifier.
- Wrapper: Selection criterion is based on evaluation by a classifier.

Concerning the types of instance selection methods the literature distinguishes between two different processes which we explain in more detail: **prototype selection(PS)** and **training set selection (TSS)**.
The term "prototype selection" is linked with the advent of instance-based or lazy learning methods [GLH15] (see definition of Nearest Neighbor Classifier in subsection 2.2.1). PS are utilizing instance based classifiers to find training sets that offer best classification performance and data reduction rates and thus to perform instance selection.
TSS methods conform to the general idea of instance selection since they can be applied to any predictive model (no limitation to instance based classifiers) [GLH15]. Figure 2.3 and Figure 2.4 illustrates the basic process for both instance selection types.

Within [WM00] the authors defined criteria to compare different training set reduction algorithms. From our perspective some of these criteria also illustrate the main objectives and functions of instance selection:

- Speed increase: Smaller training sizes result in faster training times, or in case of PS methods faster prediction of instances.
- Increase generalization accuracy: Size of training set is reduced without reducing of the generalization accuracy, in some case generalization accuracy can even increase with reduction of instances.
- Noise tolerance: Removal of certain instances can lead to simple decision boundaries, which could prevent over-fitting, but also could lead to decreasing accuracy.
- Reduction of storage: For PS methods, less training instances require less storage space.

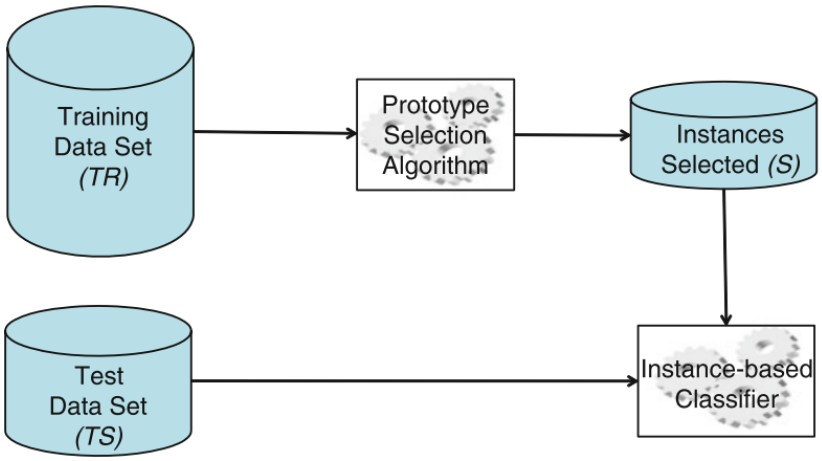


Figure 2.3.: Process of instance selection based on prototype selection, copied from [GLH15].

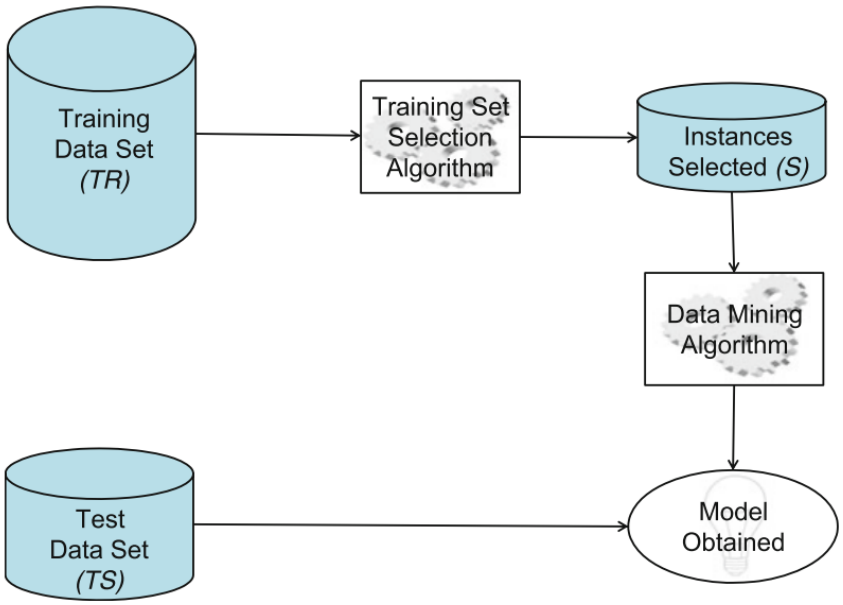


Figure 2.4.: Process of instance selection based on training subset selection, copied from [GLH15].

A similar topic to instance selection is outlier detection which we focus on in the next sub-section. A significant difference between these two topics is, that instance selection normally operates on already noise free data, whereas outlier detection is also used to remove noisy data.

2.2.4. Outlier Detection

Outlier detection attempts to find patterns in data that do not match the expected normal behavior[CBK09]. Following [HA04] ”an outlying observation, or **outlier**, is one that appears to deviate markedly from other members of the sample in which it occurs”.
[ANH16] defines the main goal of outlier detection as ”to find observations which deviate so much from other observations of the same datasets as to arouse suspicious that it was generated by a different mechanism”.
Often these outliers are the result of exceptional conditions in sensors, measurement equipment, or human errors. Within the literature there exists a plethora of synonyms and related topics for outliers and outlier detection. Therefore we want to shed some light on the various terms involved. Following [HA04] the listed terms are used in the context of outlier detection:

- anomaly detection
- deviation detection
- exception mining
- noise detection
- novelty detection

[CBK09] refers to **novelty detection** as a topic related to outlier detection, which aims at detecting previously unobserved (novel) patterns in data. The main difference is that novel patterns are typically incorporated into the normal model after being detected, in contrast to outliers.
[Pim+14] elaborated a similar delimitation between novelty detection on the one hand and outlier and anomaly detection on the other hand focusing on the different meaning of ”normal data”. The authors state that anomalies or outliers often refer to irregularities or noisy events in otherwise “normal” data, which has to be removed from data before analysis can be performed.

Noise detection or **noise removal** and **noise accommodation** are only related topics, which are dealing with unwanted noise in data, which is seen as hindrance in data analysis and needs to be removed or models need to be immunized against their harmful influence.
Throughout the rest of this work we use the terms outlier, anomaly and novelty as synonyms, since all terms refer to some kind of disturbance in the underlying process that created the data, which resulted in deviating behavior of an instance. This deviating behavior represents new, relevant and therefore valuable information, that should be incorporated into a model.
In contrast to this valuable information, we use the term "noise" for actual errors, which can potentially be harmful for further analysis and thus should be removed from the data. Although the previously listed areas of research differ in some respects, they nevertheless resort to identical techniques for identifying outliers in order to fulfill their problem statement.

Outlier detection types

In general we distinguish three different types of outliers, namely point outliers, contextual outliers, collective outliers. We shortly summarize the different types of outliers based on [CBK09].
A **point outlier** is an individual data instance that can be considered as anomalous given the rest of the data. For example, if the typical amount spent in credit card transaction for a customer is hundred Euros, an amount spend of ten thousand Euros is considered an outlier.
A **contextual outlier** is a data instance that is an anomaly only given a specific context, also referred to as conditional outlier. For example, 25 degree Celsius in summer are considered normal, while this is not the case in the mid of December. The applying contexts have to specified as part of the problem formulation. Contextual outliers are often used in spatial or temporal data.
Collective outliers are a collection of somehow related data instances which are anomalous with respect to the other data instances of a dataset. An individual instance within a collection of outliers may not be considered an outlier, but their occurrence together is considered anomalous. For example, having a very high heart rate for just a few seconds would not be considered

as outlier, but having a very high heart rate for hours would be considered a collection of outliers.

Outlier detection application areas

Outlier detection is applied in many different areas. [CBK09] provide an extensive overview of application areas for outlier detection. We only present a brief summary of the different application areas:

- Intrusion Detection: Detection of malicious activity in computer or computer network related systems
- Fraud Detection: Detection of criminal activities in banks, insurances, stock market, credit cart companies
- Medical and Public Health Anomaly Detection: Detection of anomalies concerning patient conditions, or detecting disease outbreaks
- Industrial Damage Detection: Detection of industrial machines and products to prevent escalation and losses due to problems in production
- Image Processing: Detection of changes in images over time, or abnormal regions in static images
- Anomaly Detection in Text Data: Detection of novel topics, events or news in collections of documents
- Sensor Networks: Detection of anomalies, such as faulty sensors, or abnormal events in wireless sensor networks

There are various other domains, for which the interested reader may refer directly to [CBK09].

Outlier detection categorization

[HA04] offers a categorization of different methods, based on research areas in which they have been developed. This classification includes the following areas: statistic techniques, neural networks techniques, machine learning techniques, and hybrid systems. In the very same work the authors presented three fundamental approaches concerning the problem of outlier detection based on the underlying approaches:

- Type 1: Determine outliers with no prior knowledge of the data analogous to unsupervised clustering methods
- Type 2: Model both normality and abnormality, analogous to supervised classification
- Type 3: Model only normality or in a very few cases model abnormality, which is in fact novelty detection, where soft bounded algorithms can estimate the degree of "outlierness".

An extension of these approaches from [CBK09] provides a more comprehensive list of approaches based on the underlying techniques involved:

- Classification based
- Clustering based
- Nearest Neighbor based
- Statistical
- Information Theoretic
- Spectral

We briefly summarize these approaches within the next paragraphs based on [CBK09].
Classification based approaches work analogous to classification: Within a training phase a classifier is trained with labeled trained data, and within the test phase an instance is classified as normal or outlier instance.
Clustering based techniques are based on the paradigm that unlabeled instances are assigned to clusters. There are different clustering based approach based on the underlying assumptions concerning outliers. The first category considers an instance as an outlier if it is not assigned to a cluster. The second category considers instances which are far away from their cluster-centroids as outliers. And the third category is based on the assumption that normal instances belong to large and dense clusters, whereas outliers belong to small and sparse clusters.
Nearest Neighbor Based techniques are based on the assumption that normal instances occur in dense neighborhoods, while anomalies are located in sparse areas, or are far away from their nearest neighbor.
Statistical techniques are analyzing the location of instances given the probability regions, where normal instances are located in high probability regions and outliers occur in low probability regions.
Information Theoretic techniques are dealing with the information content

by using theoretic measures, for example entropy, or mutual information. The key assumption of these techniques is that outliers change the information content of a dataset.
Spectral techniques enforce change in data representation, for example dimensionality reduction, to embed data into lower dimensional subspace in which normal instances are significantly different then outliers.

Another categorization of approaches for outlier detection arouses if the different approaches are categorized based on the underlying type of algorithm [HA04]:

- distance-based
- set-based
- density-based
- depth-based
- model-based
- graph-based

2.3. State of the Art

2.3.1. State of the Art Analysis

For the background related to instance selection we had 11 recommendations and found 2 references in this recommendations. Out of this 13 publications we finally used 5 publications for our background section and one publication is used as the reference application for a state of the art instance selection algorithm (see LDIS algorithm in Section 4.3.2).
In order to find outlier detection algorithms that were used for instance selection, we performed a state of the art (SOTA) analysis. We used the following search terms and combinations in each of the earlier presented platforms (see Section 2.1):

- ”outlier|novelty|anomaly detection” AND ”instance selection”
- ”outlier|novelty|anomaly detection” AND ”prototype selection”

- ”outlier|novelty|anomaly detection” AND ”training set | trainingset selection”

These searches resulted in 28 papers which we considered relevant for closer inspection. After closer inspection 22 papers where considered not relevant for the following reasons:

- Six papers were pure instance selection papers with no relation to outlier detection
- One paper simply applied instance selection before outlier detection
- One paper applied outlier detection in adaptive protocols without any instance selection
- Fourteen papers used totally different approaches, or had no connection of outlier detection and instance selection:
  - Three papers were used for meta learning, multiple instance learning or used mutual information selection for forecasting
  - Four papers applied noise removal instead of instance selection
  - Two papers applied sampling techniques for instance selection
  - Five papers applied preprocessing to streaming data

We provide a detailed table of these 22 irrelevant papers in Section A.1 in the Appendix.
The 28 search results also included three papers that looked at the topic from the perspective of how to use instance selection for outlier detection, see [Li11], [Li+09a], [Li+09b].
To give an example for a search result which was not suitable for our approach, we will briefly describe the following work: [VA17]. The authors proposed a novel approach for instance selection which consisted of two steps. First unrepresentative (or outlier) instances are removed from the training set using ”data editing” (a variant of instance selection)) and then the authors perform ”instance reduction” based on compression via minimum description length principle. Although this approach involves all the relevant topics it does not perform outlier detection for instance selection.
Finally this results in only two relevant results of our SOTA analysis which we will inspect in more detail in the following sub-section.

2.3.2. Outlier Detection for Instance Selection

The most promising paper of our SOTA analysis was [PC14], which is closely related to [PC12] since the authors are identical. The authors state that instance selection can be carried out via outlier detection techniques. Their approach is based on local kernel regression, and accounts for the local structure of the data space, by not only considering the distance between a point and its neighbors, but also the distance between neighbors. The basic principle of the underlying work is that the reconstruction error of an inner point is smaller than that of a boundary (or outer) point. Based on this phenomenon the "outlierness" of points is calculated. To get the estimates of "outlierness" it is measured for each point X_i how well its neighbors can estimate every feature value of X_i. For a clearer understanding of the used algorithm we depicted it in Listing 1.

Algorithm 1 Local Kernel Ridge Regression

```
Provide l (number of outliers to be removed)
m = 0 (number of outliers removed)
O = Empty list (list of removed outliers)
Create neighbor graph G and kernel matrix K from dataset D
while m != l do
    Estimate local kernel regression for each instance X_i
    Calculate reconstruction error RE_i for each X_i
    Sort all X_i based on RE_i in descending order
    Insert X_i with highest RE_i into O
    Remove X_i with highest RE_i from D
    m = m + 1
    Re-create G and K from D
end while
```

In addition to the overall algorithm, we briefly explain the most essential steps of estimation and calculation of the reconstruction error. Since the estimation is carried out using local kernel ridge regression, we briefly explain, the basic concept of it, keeping the notation used in [PC14]. Kernel ridge regression uses kernel functions to model nonlinear regression. Regression in general is given by training data $(x_i, y_i)_{i=1}^{N}$, where x_i corresponds to the

input and y_i to the target value, and estimates the targets by the given inputs. Kernel ridge regression can be defined as:

$$g(x) = \sum_{i=1}^{N} \alpha_i \mathcal{K}(x, x_i), \tag{2.1}$$

where $\mathcal{K}(.,.)$ is a kernel function and α_i are the coefficients, which are estimated by the following objective function:

$$\min_{\alpha} ||K\alpha - y||^2 + \gamma \alpha^T K \alpha, \tag{2.2}$$

where $\alpha = [\alpha_1, ..., \alpha_N]^T$, $y = [y_1, ..., y_N]^T$, γ is a small positive regularization parameter and K is the kernel matrix with dimensions N×N with $k_{i,j} = \mathcal{K}(x_i, x_j)$. The solution of 2.2 is than given by:

$$\alpha = (K + \gamma I)^{-1} y, \tag{2.3}$$

where I is an N×N identity matrix. Finally g(x) can be given as

$$g(x) = k_x^T (K + \gamma I)^{-}1 y, \tag{2.4}$$

where $k_x = [\mathcal{K}(x, x_1), ..., \mathcal{K}(x, x_N)]^T$ is a vector containing the kernel functions for instance x and x_1 to x_N.
To estimate the value of the r-th feature of an instance x_i by the corresponding r-th feature values of the neighbors of x_i **local kernel regression** is used. More formally, given a training instance x_i and $x_j, l_{rj_{x_j \epsilon N_i}}$, where N_i are the neighbors of x_i, x_j is on of this x_i and l_{rj} is the r-th feature of this neighbor. The estimation of l_{ri} with the local kernel ridge regression model for x_i is than given by:

$$g_{N_i}(l_{ri}) = k_{N_i}^T (K_{N_i} + \gamma I)^{-1} l_{rN_i} \tag{2.5}$$

where $k_{N_i}^T$ is a vector containing the kernel function for x_i and its neighbors. K_{N_i} is the kernel matrix for the kernel functions between the neighbors of x_i N_i, for example $K_{N_i} = [\mathcal{K}(x_m, x_n)], x_m, x_n \epsilon N_i$ and I corresponds to an N*N identity matrix.
What is missing is the explanation of the calculation of the **reconstruction error**, which is given by the square of the difference between the estimated

value of the r-th feature of the local kernel ridge regression model and the real value of l_{ri}, or as equation:

$$Re(l_{ri}) = (g_{N_i}(l_{ri}) - l_{ri})^2 \tag{2.6}$$

Since the values for each feature may be different, we need to normalize $Re(l_{ri})$. We therefore calculate the diagonal matrix $\mathbf{D}$ $diag(D_{11}, ..., D_{kk}))$ which is given by:

$$D_{ii} = \frac{1}{|N_i|} \sum_{x_j \epsilon N_i} (x_i - x_j)^2 \tag{2.7}$$

Additionally a weighted variance $V_w(r)$ for each feature is estimated, as:

$$V_w(r) = \sum_{i=1}^{N} (l_{ri} - \mu_r)^2 D_{ii} \tag{2.8}$$

In Equation 2.8 μ_r is the weighted mean of the r-th feature and is given by:

$$\mu_r = \sum_i l_{ri} \frac{D_{ii}}{\Sigma_i D_{ii}} \tag{2.9}$$

Finally the normalized reconstruction error RE_i is given by:

$$RE_i = \sum_{r=1}^{M} \frac{Re(l_{ri})}{V_w(r)} \tag{2.10}$$

3. Method

3.1. Basic Principle of our Approach

To introduce our approach we consider it very helpful to first introduce the general or common approach to feature selection in combination with classification (also applicable to regression problems). This general approach is depicted in Figure 3.1. The typical sequence starts by dividing available data into disjunct training and test sets. The training set is then provided to a specific feature selection algorithm, which filters the training set for the most relevant features. The training set with only the selected features is then provided to a classification algorithm which builds a model to predict the output for the given input. After the model is build the test set is also filtered for the only selected features and this filtered test set is used to evaluate the performance of the trained model.

The difference of our approach is that we filter the instances of the training set for only "relevant" instances before making these selected instances available to a feature selection algorithm. We present our approach in Figure 3.2. Due to the pre-selection of training instances, this should positively influence the feature selection algorithm (for example, other features are considered relevant as with the generic approach).

The selection of the most relevant instances in the training set requires an algorithm that assigns a specific score to each training instance. Based on these scores, the training instances are sorted and only the top n instances are selected. We further describe this concept in more detail in the following section.

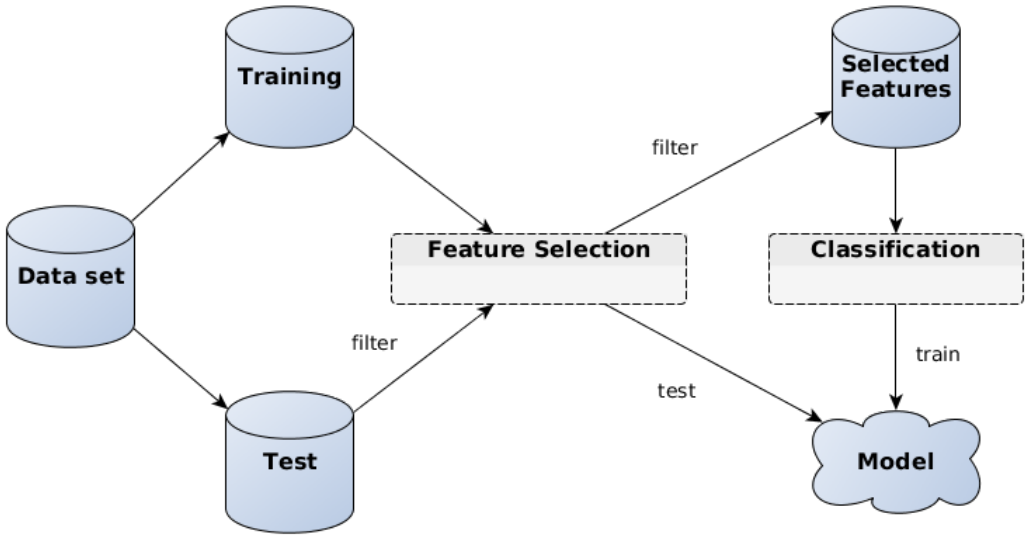


Figure 3.1.: Figure shows the general approach and sequence for feature selection combined with classification.

3.2. Concepts

For the instance selection we are interested in discriminant and thus relevant instances, as we expect a more specific selection of features. To achieve this goal we are guided by the simple but elegant principle of nearest neighbor based outlier detection techniques: Normal data instances occur in very dense neighborhoods, whereas outliers occur in sparse neighborhoods [CBK09]. If we transfer the principle of density to that of distances, we can simplify this principle by imagining that an outlier is farther away from his nearest neighbors than a normal instance that has many neighbors within a short range. This difference in neighborhood is also visualized in Figure 3.3.

This distinction due to the density or distance within a neighborhood represents a direct link between instance selection and outlier detection research areas. In order to implement our approach we have to adapt the algorithm presented in Section 2.3.2 as follows: Rather than performing the main loop

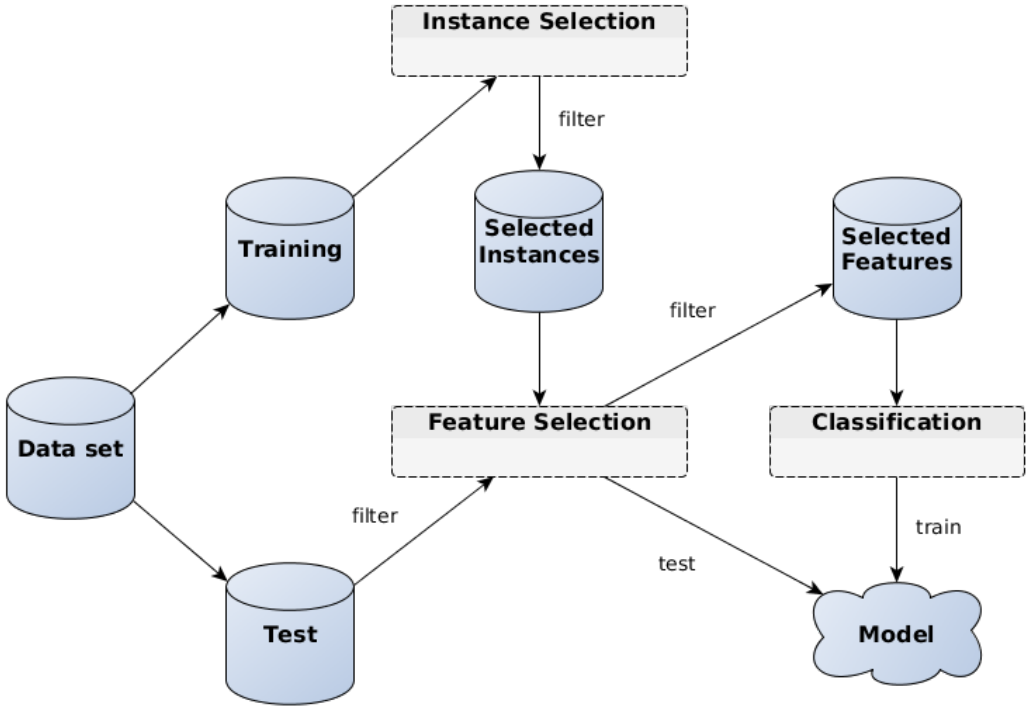


Figure 3.2.: Figure shows the general approach and sequence for instance selection for feature selection combined with classification.

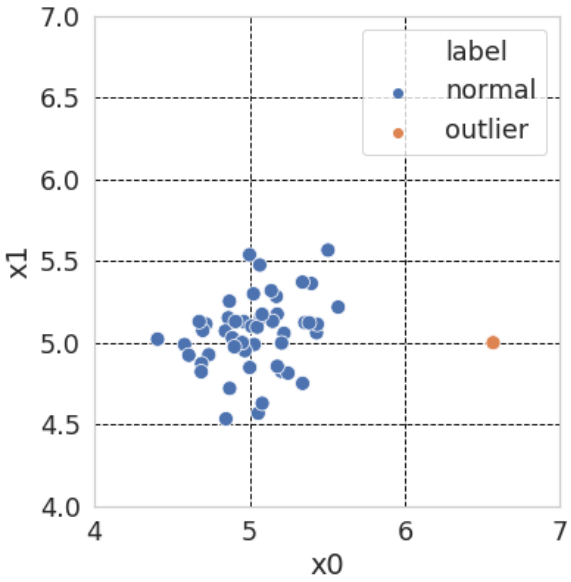


Figure 3.3.: Figure shows a dense cluster of normal points with low distances to the nearest neighbor for the normal instances and a high distance for the outlier on the right.

"m-times" (for removing m outliers) we only need to calculate the total reconstruction error RE_i for each instance once, sort all instances based on their RE_i values and then select the top n instances to filter our training instances for the following feature selection. Since we have a criterion independent of the evaluation of a classifier, this approach is a typical filter based instance selection approach. Further, as we focus on selecting training instances (for feature selection though), our approach is a typical training set selection algorithm.

For further investigation how instance selection affects the feature selection and consequently the performance of the classification, we first designed our experiments. For this first orientation we used the Waikato Environment for Knowledge Analysis, short **Weka** [FHH16]. Weka is a machine learning suite written in Java, which can be used with a graphical user interface (gui) and also programmatically via a well documented API [1]. After the first manual experiments via the Weka gui it soon became clear that due to the many different components and their parameters, manual experiments were by no means an option. So we decided to build our own pipeline that allowed us to do a large number of different experiments in a systematic way. We were able to draw on a basic concept, which we had developed in the course of another research project [Sta+19]. Due to changed requirements we had to adapt the pipeline components but not the two main underlying principles. The first main principle we had to follow was modularity. In doing so, we wanted to ensure that the different components (for example, outlier detection and feature selection) could exchange information in a defined manner without creating specific dependencies between these different components. Firstly, it called for a so-called context with which information could be passed on from one component to the next, and second, general methods for uniformly accessing these components. These issues were addressed with the general components `PipelineContext` and the interface `PipelineStep` (see details in section 3.3.1.
The second main principle we were following was interchangeability. By following this principle we wanted to guarantee that the different variants of components (for example different algorithms used for feature selection) are independent from all other steps and are completely interchangeable.

[1] http://weka.sourceforge.net/doc.stable-3-8/

We have also implemented this principle with the general `PipelineStep` principle, but also with so-called wrapper classes, which are also explained in more detail in Subsection 3.3.1.

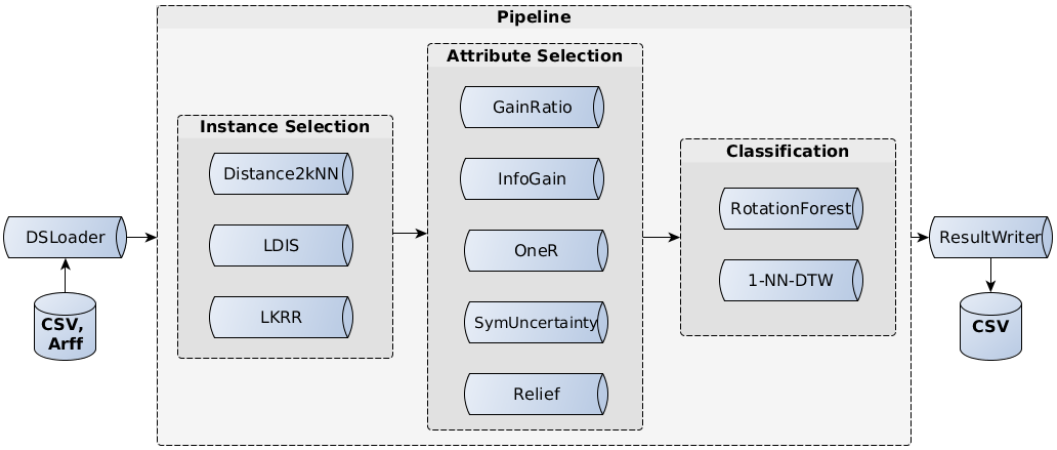


Figure 3.4.: The different component types with its specific instances of our pipeline.

3.3. Implementation

In this section we describe all components and their parameters of our developed pipeline. We would like to point out that the default terminus for feature in Weka is ”attribute”. The implication of this is that all feature selection components in Weka are consequently referred to as ”attribute selectors”. In order to avoid confusion we will also use the term ”attribute selection” when it comes to components of Weka or our developed pipeline. In the final Subsection 3.3.2 of this chapter we also explain the requirements and the instruction for installing and executing of our pipeline.

3.3.1. Components and Parameters

Pipeline Components

`PipelineStep` represents a simple interface which only defines the two functions `performStep` and `printPipelineStepDetails()`, where the later only defines a method for logging the basic members for this `pipelineStep`. The `performStep` function is the main entry point to perform a defined pipeline step. To load and store data the only parameter is the `PipelineContext` of the current pipeline.
A `PipelineContext` is used to exchange information between different `PipelineSteps`, for example an algorithm which is used for instance selection stores these instances in the current `PipelineContext`. These instances are later loaded from the following attribute selection `PipelineStep`. The `PipelineContext` provides several generic structures for the administration and exchange of different instances, parameters, and results.
These two components are complemented by the `Pipeline` object. This is the wrapper class for a pre-defined pipeline consisting of a list of different `PipelineSteps` and a `PipelineContext`.

General Components

`DSLoader` is the main class for loading the data sets used for our experiments. We use the convention that the data files use the same prefix as the `datasetName` plus two specified prefix for the training and testing set and a corresponding file ending (for example ".csv" for csv files). After providing the fully qualified folder where the data set is stored under the provided `datasetName` to the constructor, the training or test data sets can be loaded via the corresponding (`loadTrainingDataset()`, `loadTestDataset()`) functions.
`AbstractPipelineCsvResult` is the main abstract class for all csv results and unifies functions that are applicable generally, regardless of the type of result (for example the general structure for storing fields and their corresponding values). Within our pipeline we have the following different types of csv results:

- `ClassificationCSVResult` stores the result of classification experiments (for example, the name of the used classifier, precision and recall)
- `AttributeSelectorCSVResult` stores the result of attribute selection experiments (for example, the name of the used `AttributeSelector`, number of selected attributes)
- `OutlierDetectionCSVResult` stores the result of outlier detection experiments (for example, the name of the algorithm used, number of instances selected)

These aforementioned results can be written to output files by using the `AbstractPipelineCsvResultWriter`. This class has only one method, `writeOutputFileFromResultList()` which writes the given `AbstractPipelineCsvResults` to the given `filePath`. Our pipeline uses a few auxiliary classes (for example `InstanceUtils`). For the description of these additional classes we would like to refer to the corresponding documentation in the source files of our git-repository.

Wrapper Classes

In principle, Weka already offers corresponding classes and functions for almost all requirements for our approach. To integrate these existing classes into our pipeline and better manage the numerous parameters, we had to pack these existing classes into so-called wrapper classes. This wrapping allows us to continue to comply with our main principles modularity and interchangeability mentioned in Section 3.2. We briefly describe the developed wrappers in the following sub-section.
The `AbstractNNWrapper` is the main class for instance and outlier detection algorithms. Within this class the main members and functions for all the sub-classes are implemented. These sub-classes share the following relevant parameters mapped as instance variables:

- `kNeigbours`: Number of k-nearest neighbors used for search and distance calculations.

- `odSelRatios`: Ratio which represents the percentage of instances to finally select (for example 0.1 means that 10 percent of instances are selected)
- `odDescOrder`: Whether to use descending order for sorting the instances (for example, true means use that descending order is used and false means that ascending order is used)
- `nnSearch`: Always `LinearSearch`, since Weka only supports all distance functions [2] only for the `LinearSearch` at the moment.

Furthermore `AbstractNNODWrapper` implements the `PipelineStep` function `performStep`. Within this function the sub-class specific method `performFiltering` is called, data is loaded and afterwards stored in the `PipelineContext`. The concrete sub-classes are listed bellow, and further description for them is available in Section 4.3.2:

- Distance2kNNODWrapper
- LDISWrapper
- LKRRODWrapper

`AttributeSelectorWrapper` is the wrapper class for the attribute selection algorithms. The concrete attribute selection algorithms are further described in Section 4.3.3. Based on the member variable `attributeRatio` the given ratio of attributes from the training data set are selected (for example 0.1 means that 10 percent of the attributes are selected). The actual number of attributes to select is calculated via the method `getNumToSelectFromRatio` and takes the number of attributes without the class attribute as argument. In case `getNumToSelectFromRatio` returns 0, we override the value to one, since we always want to have at least one best attribute to be selected for further processing. Once the attributes are evaluated the training and test instances are reduced to the given `attributeRatio` of attributes. These instances are then stored within the `PipelineContext` which are then used by the classification algorithms.
`AbstractClassifierWrapper` has five concrete classification algorithms as sub-classes which can be used in our pipeline. For the experiments of this work we only used `Rotation Forest` and `DTW1NN`. `AbstractClassifierWrapper` has no relevant members and parameters, Rotation Forest neither, DTW1NN

[2] We will have a closer look at these distance functions in Section 4.3.5

has only the fixed default window size for dynamic time warping which is one by default.

3.3.2. Installation and Execution of our Pipeline

We first want to point out that we provide all our source-code, data and results in a git repository [3]. Within the root folder of this repository we also provided a `README` file which contains an overview of our library. Our framework **Outfisltisl** has the following requirements:

- Java Oracle JDK version 1.8.X
- Maven version 3.3.X [4]

Further dependencies are listed in the corresponding `pom.xml` in the root folder of our project. For the reference timeseries classifiers we used the source-code from the paper [BL14] [5]. From this source code we extracted the DTW1NN and Rotation Forest classifiers and build the jar file `timeseries-classification-1.0.0.jar` which is present in the `lib` folder of our project. The specified commands refer to the execution in a Linux environment. For the installation of our pipeline the following steps are necessary:

- Requirements from above fulfilled
- Clone git-repository:
  `https://git.know-center.tugraz.at//r/ dcemernek/outfisltisl.git`
- Execute the following commands in root folder (= outfisltisl) of project:
  - `mvn initialize` (required only once)
  - `mvn package` (each time changes are made within source code)
- After execution of `mvn package` there should be a folder "target"
- Folder "target" should contain jar file "outfisltisl-jar-with-dependencies.jar"

For the execution of the experiments the following steps must be executed:

- Requirements: Installation from above fulfilled

[3] https://git.know-center.tugraz.at/summary/?r=~dcemernek/outfisltisl.git
[4] Maven is a tool to build source code and manage dependencies
[5] see also: https://bitbucket.org/TonyBagnall/time-series-classification

- Again given commands refer to a Linux environment
- Application can be started with command:
  `java -jar outfisltisl-jar-with-dependencies.jar`
- This command also prints the Help information including relevant parameters to use.
- Example command that starts the outlier detection experiments for dataset FordA in folder "../data/datasetsExp" using Rotation Forest Classifier and LKRR as instance selection algorithm:
  `java -jar outfisltisl-jar-with-dependencies.jar -i ../data/datasetsExp/ -d FordA -e od -c RotF -o LKRR`

4. Evaluation

4.1. Recap

As a quick refresher, we briefly review our main research question: Is it possible to positively increase the performance of learning algorithms by providing only a specific subset of our training instances to feature selection algorithms? In order to evaluate this question and thus our approach we need to define and describe the following components in this section:

- Datasets to test our approach
- An outlier detection algorithm to perform instance selection to filter training instances
- Additional instance selection algorithms to compare our approach to
- Feature selection methods to perform feature selection on our filtered training instances
- Classification algorithms to measure potential performance influence of our approach

Before we give a detailed description of the above components in Section 4.3, we start with a general introduction on how to evaluate time series classification.

4.2. Evaluation of Time Series Classification

For the current state of the art concerning the evaluation of time series classification we will focus on the following work "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances" [Bag+17]. In this paper the authors implemented 18

recently proposed algorithms and compared them against two benchmark classifiers. In this evaluation they authors found out that only 9 of this 18 algorithms were significantly more accurate than the benchmark classifiers. They authors published all used datasets and detail results on a dedicated website, namely the UEA & UCR Time Series Classification Repository[1]. Since the datasets, algorithms and results are constantly extended we decided to use the same standard benchmark classifiers and datasets from this site. Furthermore by only using public available datasets we want to support transparency within our research field.
The second piece of work we use to evaluate our approach is the somewhat older but still very relevant paper named "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration" [KK03]. In this paper the authors conducted a survey about time series data mining papers and analyzed their evaluation, underlined by the following relevant statistics per paper:

- Size of test datasets - Median only 10,000 test data
- Number of rival methods - Median number is 1 so authors only compared to one method
- Number of different test datasets - On average each contribution on 1.85 datasets

Since we are mostly dealing with Industrial Damage Detection, which involves sensor data, we focus on datasets of this type. The largest dataset for sensor data has only 9236 instances, we can not address the first point, but the authors of [Bag+17] already addressed the importance of a better representation of larger datasets in the future.
For the number of rival methods we refer to the corresponding Sub-section 4.3.2, we only state that we evaluated a plethora of different combinations of outlier detection, feature selection and time series classification algorithms. Concerning the number of different test datasets we can clearly state that by using 8 different sets from the UEA & UCR Time Series Classification Repository in total. The datasets and all other components of our experiments are described in the following section.

[1] http://timeseriesclassification.com

Dataset	Train Size	Test Size	#Features	#Classes
Earthquakes	322	139	512	2
FordA	3601	1320	500	2
ItalyPowerDemand	67	1029	24	2
Lightning2	60	61	637	2
MoteStrain	20	1252	84	2
SonyAIBORobotSurface1	20	601	70	2
SonyAIBORobotSurface2	27	953	65	2
Wafer	1000	6164	152	2

Table 4.1.: Table shows the selected datasets and their attributes.

4.3. Experimental Design

4.3.1. Datasets

For evaluation of our contribution we use datasets from the UEA & UCR Time Series Classification Repository relaunched 2018 with 128 datasets [Dau+18]. For this contribution we stick to the following criteria to select datasets:

- Type of dataset: Sensor, since Industrial Damage Detection mostly deals with this kind of data
- Number of classes: Two, since most of the time we need to distinguish between normal and abnormal data
- Evaluation results for comparison available on UEA & UCR TSC website

These criteria resulted in the datasets described in Table 4.3.1. Since the following datasets have no results for comparison we could not include them: DodgerLoopGame, DodgerLoopWeekend, FreezerRegularTrain, FreezerSmallTrain. Additionally we excluded the FordB dataset since it is too similar to the FordA dataset. In contrast we included both the SonyAIBORobotSurface1 and the SonyAIBORobotSurface2 dataset since there are very different.

The datasets from the UEA & UCR TSC Repository are typically already split into a training and a testing set and all the datasets are z-normalized. At first it seems cumbersome to use prefixed training and test sets, but this has the reason, that some of the datasets are designed so the train test split removes bias. Combining the different sets again for performing cross validation re-introduces this bias. Furthermore that authors state that "it is not feasible to cross validate everything" [BL14]. We also want to draw the readers attention to this very sound work that critically deals with cross-validation in time series analysis [BHK18].

4.3.2. Algorithms for Instance Selection

In the course of the evaluation of our approach we have used the following algorithms for instance selection:

- Distance2kNN: Distance to k-Nearest-Neighbors
- LDIS: Local density-based instance selection
- LKRR: Local kernel ridge regression for instance selection

Distance2kNN is based on the idea which is summarized in [CBK09], were various authors (Eskin et al. [2002], Angiulli and Pizzuti [2002] and Zhang and Wang [2006]) calculated an anomaly score for each instance based on the sum of its distances to their k nearest neighbors. Since this algorithm is very simple and quick to implement, it served as a feasibility study and baseline for our approach. Additionally it is a typical representative of a pure distance based outlier detection algorithm.
LDIS is a typical state of the art instance selection algorithm, which evaluates instances of each class separately, preserving only the densest instances in a certain neighborhood [CA15].
LKRR is the only relevant result of our state of the art analysis for the usage of outlier detection algorithms explicitly for instance selection and was already described in 2.3.2.

4.3.3. Feature Selectors

For the experiments, we generally want to point out that we are using feature selection only and do not change the representation (= feature construction) of the data which we evaluate during instance or feature selection.
Following [AAB11] we focused on feature selection methods which select the most relevant features as a a combination of an individual performance measure and a so called cutting criteria (for example, fixed number of features or ratio of features to select) to select the best number of features. All of the listed feature selectors are filter based feature selection methods. OneR feature selector is the only exception since it is using the OneR classifier to select the best subset of features. We are using the following feature selectors, which are already provided by Weka (Source: [AAB11] and Weka Doc):

- **GainRatio** [2]
  - Gain ratio is the ratio between information gain and the entropy of the feature
  - GainR(Class, Attribute) = (H(Class) - H(Class | Attribute)) / H(Attribute))
- **InfoGain** [3]
  - Also known as **Mutual Information**, measures the worth of an feature with the corresponding class
  - InfoGain(Class,Attribute) = H(Class) - H(Class | Attribute)
- **OneR** [4]
  - OneR is a single variable classifier used for feature selection
  - Uses the minimum-error attribute for prediction a given class
- **ReliefF** [5]

[2] http://weka.sourceforge.net/doc.stable-3-8/weka/attributeSelection/GainRatioAttributeEval.html
[3] http://weka.sourceforge.net/doc.stable-3-8/weka/attributeSelection/InfoGainAttributeEval.html
[4] http://weka.sourceforge.net/doc.stable-3-8/weka/attributeSelection/OneRAttributeEval.html
[5] http://weka.sourceforge.net/doc.stable-3-8/weka/attributeSelection/ReliefFAttributeEval.html

  - ReliefF evaluates individual features, but also takes into account the relation among features, by considering the value of a given attribute also by its nearest instance of the same and different class

- **SymmetricalUncertainty** [6]
  - Evaluates the worth of an attribute by measuring the symmetrical uncertainty with respect to the class.
  - SymmU(Class, Attribute) = 2 * (H(Class) - H(Class | Attribute)) / H(Class) + H(Attribute).

4.3.4. Classifiers

Since our work is not about the development of a new TSC algorithm, we will follow the recommendation of [Bag+17] for the usage of DTW1NN and Rotation Forest as base line algorithms. We already provided a brief description of these algorithms in subsection 2.2.1. In the course of time series analysis and in particular time series classification, a suitable data representation plays an important role (compare sub-section 2.2.1), therefore with the DTW1NN we decided for a classifier, which transforms given data representation by means of performing dynamic time warping.

4.3.5. Parameters and Experiments

For the instance selection we defined the following value ranges for the parameters and instances of components:

- kNeigbours: 1, 5, 10
- odSelRatios: 0.0005, 0.05, 0.25, 0.5, 0.9
- odDescOrder: true, false
- nnSearch: LinearSearch

[6] http://weka.sourceforge.net/doc.stable-3-8/weka/attributeSelection/SymmetricalUncertAttributeEval.html

- DistanceFunctions: ChebyshevDistance [7], EuclideanDistance[8], ManhattanDistance[9], MinkowskiDistance[10]

For the feature selection we defined the following value ranges for the parameters and instances of components:

- attSelRatios: 0.002, 0.01, 0.1, 0.33, 0.66
- AttributeSelectors: GainRatio, InfoGain, OneR, Relief, SymmatricalUncert

For both the odSelRatios and attSelRatios we followed the datasets with the largest number of instances and features. The value 0.002 depends on the dataset with the largest number of features (Lightning2 with 637 features). Hereby we want to force that for each dataset at least for the attribute selection only one feature is selected. For the instance selection we wanted to have at least two instances as output of the instance selection.
Based on the different characteristics of our parameters and components, such as distance functions, there are 3,000 experiments per dataset and classifier. This number is the product of:
num kNeighbors * num odSelRatios * num odDescOrder * num Search * num distanceFunctions * num attSelRatios * and num AttributeSelectors
given by: 3 * 5 * 2 * 1 * 4 * 5 * 5 = 3,000.
These 3,000 experiments were performed for each combination of dataset, classifier and instance selection method. For clearer differentiation we will refer to these different combinations as **experimental suites**. For example all experiments on dataset Earthquakes with Rotation Forest combined with the LKRR instance selection method are one experimental suite.
Since we have eight different datasets, three different instance selection methods, and two classifiers we have 48 different experimental suites or 144,000 experiments in total.

[7] http://weka.sourceforge.net/doc.stable-3-8/weka/core/ChebyshevDistance.html
[8] http://weka.sourceforge.net/doc.stable-3-8/weka/core/EuclideanDistance.html
[9] http://weka.sourceforge.net/doc.stable-3-8/weka/core/ManhattanDistance.html
[10] http://weka.sourceforge.net/doc.stable-3-8/weka/core/MinkowskiDistance.html

4.3.6. Performance Metrics

Within this section we briefly discuss the metrics we used to measure the performance of the classifiers. The results on the UEA & UCR TSC Repository are only available as **TP-rate** (or accuracy). Since we need to compare our results to the baseline of the UEA & UCR TSC Repository, we also use the TP-rate as a metric to measure the performance of the classifiers.
The TP-rate computes the fraction of examples that are correctly classified. It is one of the most used evaluation metric for classification tasks. Nevertheless this metric has the fundamental problem that rare classes are neglected in contrast to normal classes. The usage of more comprehensive classification metrics, for example the F-Measure or the geometric mean was already suggested in 2004 by [Wei04].
Since we are aware of this problem, we listed for each classifier and each dataset the local baseline, the best feature selection result and the best instance selection result in Section A.2 in the Appendix.

4.3.7. Environment and Experiment Preparation

In order to inspect the detailed results or even re-run our experiments, we would like to refer to our public git-repository [11].
In order execute the experiments the requirements in Section 3.3.2 must be fulfilled. We also provided an overview of the basic components and their relations in Figure 4.1.
Our application has three different types of experiments, namely Classification, Attribute Selection and Instance Selection.
The classification experiments only perform pure classification, without any preprocessing of the data. The results of the classification experiments serve as local baseline.
Attribute selection experiments involve attribute selection steps before the classification step.
The instance selection experiments involve all steps, including the instance selection for the attribute selection followed by classification.

[11] https://git.know-center.tugraz.at/summary/?r=~dcemernek/outfisltisl.git

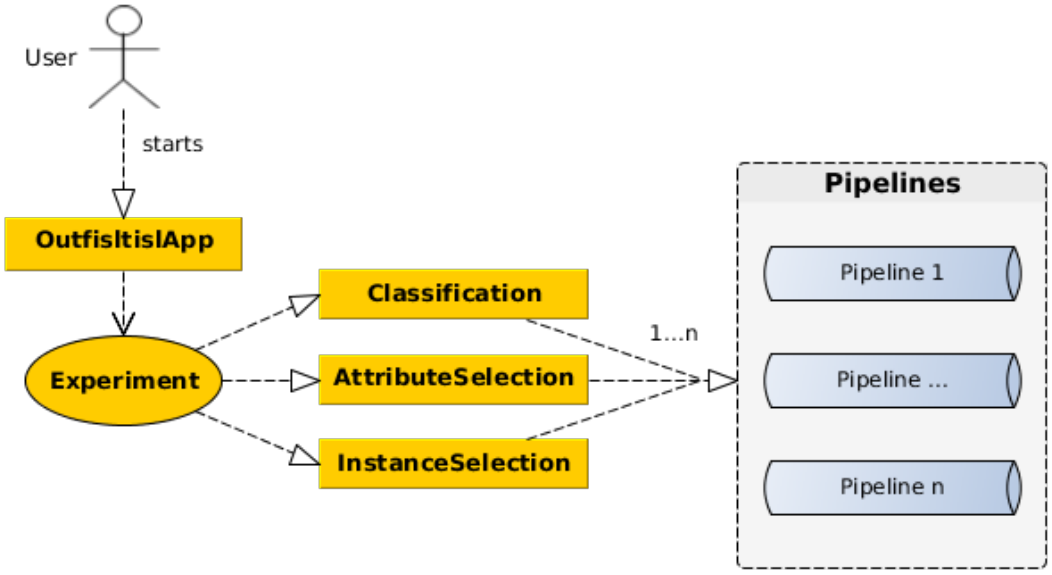


Figure 4.1.: Figures shows the dependencies and flow of a typical experiment.

4.4. Results

Within this section we present an overview of the results of our experiments. For a better overview we divided the results per classifier. For each classifier, there is a table that represents the TP-rate (rounded to three digits) for each experimental suite.
We also present the baselines for each classifier and dataset based on the UEA & UCR TSC Repository in column ”TSC” for method ”NoAS” (= no attribute selection). Unfortunately, our local baselines for the Rotation Forest differ from the results of the UEA & UCR Time Series Classification Repository, which is the reason we have also specified our local baseline in the column ”LocBase” also with the method ”NoAS”. Furthermore we included the results for attribute selection and classification indicated by method ”NoIS” (= no instance selection).
For better visibility we formatted the following items: best result per row and best method per dataset and classifier are formatted in bold.
We also provided bar charts for each dataset which can be found in the Section A.3 and an extended version of performance metrics including

Precision, Recall and F-Measure in Section A.2 in the Appendix. For the results for each single experiment please refer to our git-repository [12].

4.4.1. Results Rotation Forest

Within this sub-section we provide the results table for the Rotation Forest classifier in Table 4.4.1.

[12] https://git.know-center.tugraz.at/summary/?r=~dcemernek/outfisltisl.git

Dataset	Method	TSC	LocBase	GainR	InfoGain	OneR	Relief	SymmU
Earthquakes	NoAS	0.748	0.755					
Earthquakes	NoIS			0.784	0.784	0.799	0.777	**0.806**
Earthquakes	Dist2kNN			0.799	0.791	**0.806**	0.791	0.791
Earthquakes	**LDIS**			0.791	0.799	0.799	**0.813**	0.806
Earthquakes	**LKRR**			**0.813**	0.799	0.806	0.799	0.806
FordA	NoAS	0.845	0.749					
FordA	NoIS			**0.76**	**0.76**	0.744	0.736	**0.76**
FordA	Dist2kNN			**0.786**	**0.786**	0.769	0.76	**0.786**
FordA	LDIS			0.785	0.775	0.777	0.773	**0.786**
FordA	**LKRR**			0.778	**0.789**	0.775	0.762	**0.789**
ItalyPower Demand	NoAS	0.973	0.969					
ItalyPower Demand	NoIS			0.971	0.973	**0.974**	0.97	0.969
ItalyPower Demand	Dist2kNN			0.973	0.973	0.972	**0.975**	0.973
ItalyPower Demand	LDIS			**0.974**	0.973	**0.974**	**0.974**	**0.974**
ItalyPower Demand	**LKRR**			0.975	**0.976**	0.974	0.975	0.974
Lightning2	NoAS	0.689	0.754					
Lightning2	NoIS			**0.787**	0.77	0.77	0.754	0.754
Lightning2	**Dist2kNN**			0.82	0.82	0.82	**0.836**	0.82
Lightning2	LDIS			0.803	**0.82**	**0.82**	**0.82**	**0.82**
Lightning2	**LKRR**			0.803	**0.836**	0.82	0.82	**0.836**

Dataset	Method	TSC	LocBase	GainR	InfoGain	OneR	Relief	SymmU
MoteStrain	NoAS	0.880	0.868					
MoteStrain	NoIS			0.853	**0.887**	0.875	0.861	0.853
MoteStrain	Dist2kNN			0.884	0.885	0.894	**0.895**	0.883
MoteStrain	LDIS			0.883	**0.9**	0.894	0.895	**0.9**
MoteStrain	**LKRR**			0.902	0.897	0.902	**0.906**	0.897
SonyAIBORobot Surface1	NoAS	0.809	0.775					
SonyAIBORobot Surface1	NoIS			0.779	**0.799**	0.667	0.759	0.779
SonyAIBORobot Surface1	Dist2kNN			0.864	0.864	0.864	**0.88**	0.864
SonyAIBORobot Surface1	**LDIS**			0.864	**0.885**	0.864	0.88	**0.885**
SonyAIBORobot Surface1	LKRR			0.867	0.864	0.864	**0.88**	0.864
SonyAIBORobot Surface2	NoAS	0.808	0.779					
SonyAIBORobot Surface2	NoIS			0.811	0.805	0.807	**0.837**	0.835
SonyAIBORobot Surface2	Dist2kNN			0.839	0.835	0.837	**0.854**	0.837
SonyAIBORobot Surface2	LDIS			**0.841**	**0.841**	0.837	0.837	**0.841**
SonyAIBORobot Surface2	**LKRR**			0.845	0.844	0.837	**0.856**	0.844

Dataset	Method	TSC	LocBase	GainR	InfoGain	OneR	Relief	SymmU
Wafer	NoAS	0.994	0.992					
Wafer	NoIS			0.996	**0.997**	0.996	0.996	0.996
Wafer	Dist2kNN			0.997	**0.999**	**0.999**	0.998	0.997
Wafer	LDIS			0.996	0.998	0.998	**0.999**	**0.999**
Wafer	**LKRR**			0.995	0.999	**1**	0.999	0.999

Table 4.2.: Best TP-rates for all datasets, instance selection algorithms and attribute selectors for the Rotation Forest classifier.

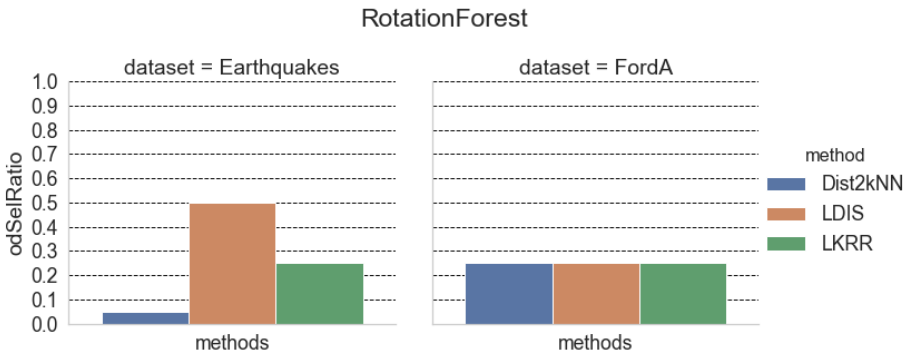


Figure 4.2.: OdSelRatio values for classifier Rotation Forest and corresponding datasets.

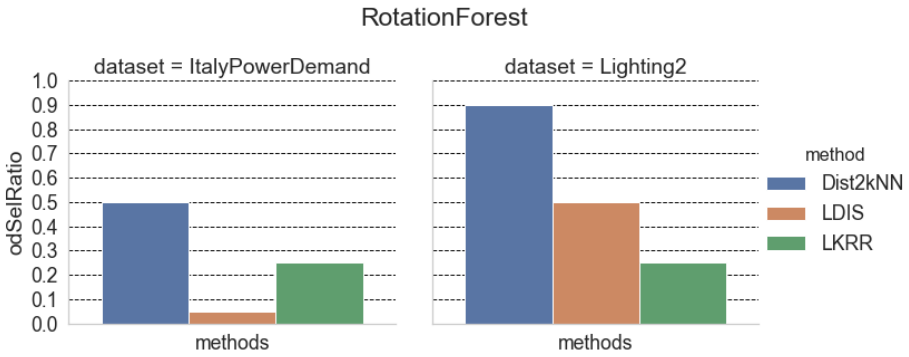


Figure 4.3.: OdSelRatio values for classifier Rotation Forest and corresponding datasets.

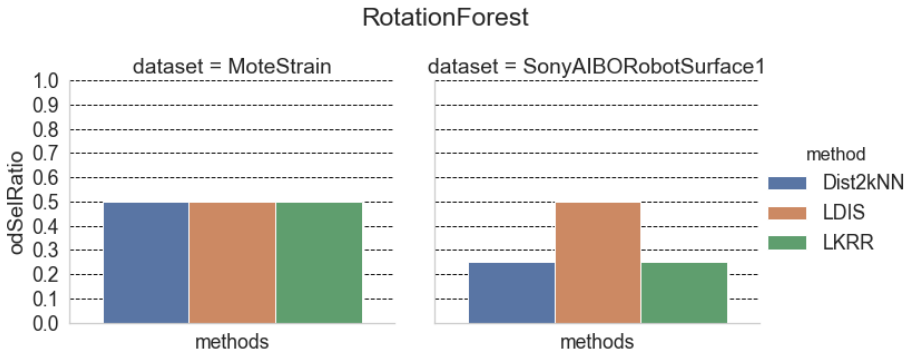


Figure 4.4.: OdSelRatio values for classifier Rotation Forest and corresponding datasets.

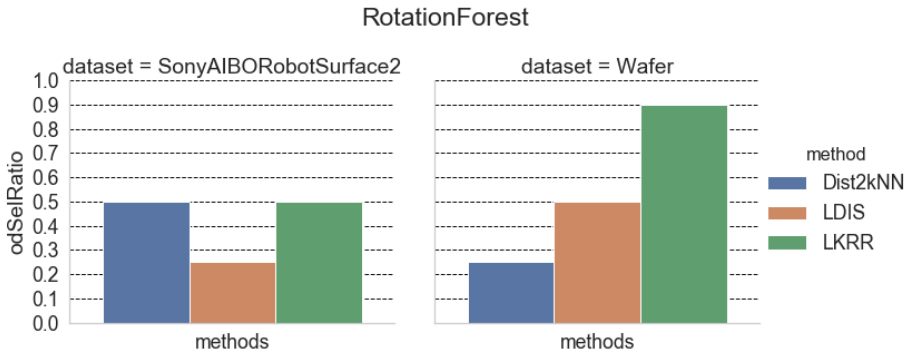


Figure 4.5.: OdSelRatio values for classifier Rotation Forest and corresponding datasets.

4.4.2. Results DTW1NN

Within this sub-section we provide the results table for the 1-Nearest Neighbor with dynamic time warping classifier in Table 4.4.2.

Dataset	Method	TSC	LocBase	GainR	InfoGain	OneR	Relief	SymmU
Earthquakes	NoAS	0.719	0.719					
Earthquakes	NoIS			0.748	**0.755**	0.705	**0.755**	**0.755**
Earthquakes	Dist2kNN			0.755	0.755	0.763	**0.77**	0.748
Earthquakes	LDIS			0.748	0.77	0.777	**0.791**	0.755
Earthquakes	**LKRR**			**0.799**	0.777	0.755	0.77	**0.799**
FordA	NoAS	0.555	0.555					
FordA	NoIS			0.613	0.613	**0.626**	0.62	0.613
FordA	Dist2kNN			**0.646**	**0.646**	0.644	0.637	**0.646**
FordA	LDIS			0.636	0.645	0.659	**0.664**	0.636
FordA	**LKRR**			0.646	0.646	**0.655**	**0.655**	0.646
ItalyPower Demand	NoAS	0.95	0.95					
ItalyPower Demand	NoIS			0.973	0.972	**0.976**	0.968	0.973
ItalyPower Demand	Dist2kNN			0.974	0.973	**0.976**	0.973	0.972
ItalyPower Demand	LDIS			0.974	**0.976**	0.975	0.974	0.971
ItalyPower Demand	**LKRR**			0.976	**0.977**	0.976	0.975	**0.977**
Lightning2	NoAS	0.869	0.869					
Lightning2	NoIS			0.803	0.738	0.803	0.77	**0.82**
Lightning2	Dist2kNN			0.803	**0.869**	0.852	0.836	**0.869**
Lightning2	**LDIS**			0.82	0.836	0.852	**0.885**	0.836
Lightning2	LKRR			0.852	**0.869**	0.836	0.836	0.852

Dataset	Method	TSC	LocBase	GainR	InfoGain	OneR	Relief	SymmU
MoteStrain	NoAS	0.835	0.835					
MoteStrain	NoIS			0.868	0.852	**0.879**	0.868	0.868
MoteStrain	**Dist2kNN**			0.853	0.862	0.875	**0.89**	0.862
MoteStrain	**LDIS**			0.863	0.881	0.883	**0.89**	0.881
MoteStrain	**LKRR**			0.883	0.872	0.886	**0.89**	0.883
SonyAIBORobot Surface1	NoAS	0.725	0.725					
SonyAIBORobot Surface1	NoIS			0.772	**0.799**	0.759	0.76	0.772
SonyAIBORobot Surface1	Dist2kNN			**0.854**	0.845	0.845	0.837	0.845
SonyAIBORobot Surface1	**LDIS**			**0.885**	0.845	0.824	0.852	0.87
SonyAIBORobot Surface1	LKRR			0.854	**0.87**	0.845	0.865	**0.87**
SonyAIBORobot Surface2	NoAS	0.831	0.831					
SonyAIBORobot Surface2	NoIS			0.816	0.813	0.834	**0.85**	0.836
SonyAIBORobot Surface2	Dist2kNN			0.853	0.852	**0.866**	0.841	0.853
SonyAIBORobot Surface2	LDIS			0.857	0.858	**0.867**	**0.867**	0.845
SonyAIBORobot Surface2	**LKRR**			0.858	0.857	**0.886**	0.858	0.858

Dataset	Method	TSC	LocBase	GainR	InfoGain	OneR	Relief	SymmU
Wafer	NoAS	0.98	0.98					
Wafer	NoIS			0.982	0.982	0.98	**0.991**	0.979
Wafer	Dist2kNN			0.991	**0.994**	0.992	**0.994**	0.99
Wafer	**LDIS**			**0.995**	0.991	0.99	0.994	0.992
Wafer	**LKRR**			0.994	**0.995**	0.992	**0.995**	0.994

Table 4.3.: Best TP-rates for all datasets, instance selection algorithms and attribute selectors for the DTW1NN classifier.

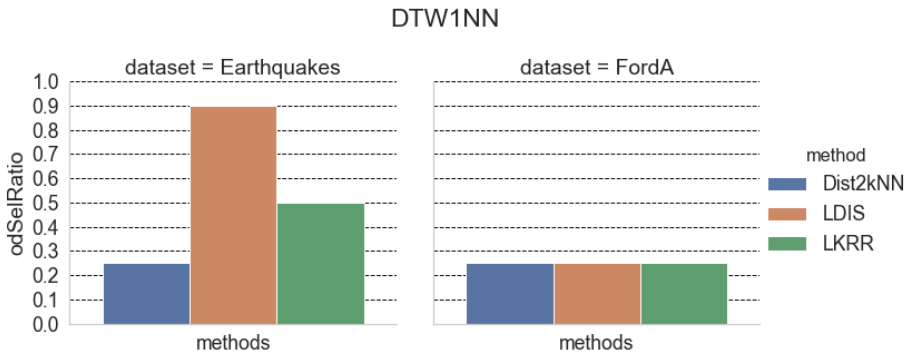


Figure 4.6.: OdSelRatio values for classifier DTW1NN and corresponding datasets.

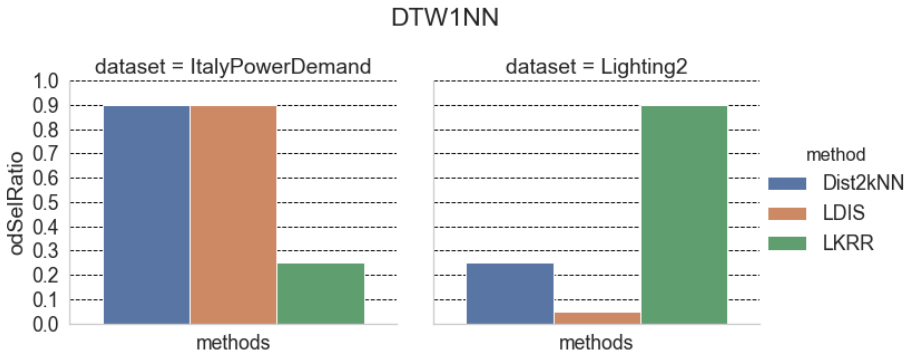


Figure 4.7.: OdSelRatio values for classifier DTW1NN and corresponding datasets.

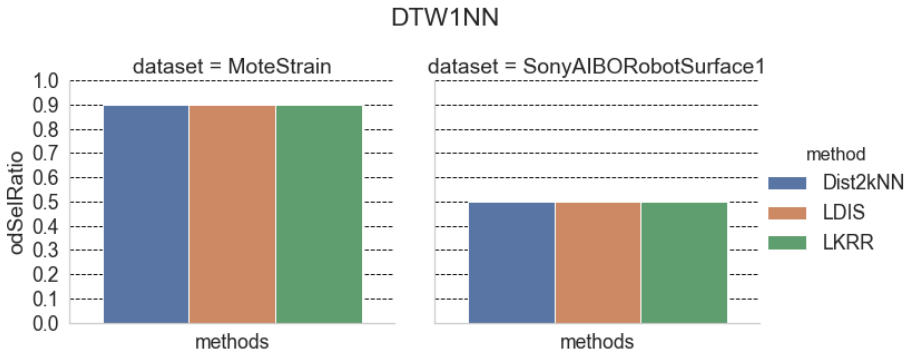


Figure 4.8.: OdSelRatio values for classifier DTW1NN and corresponding datasets.

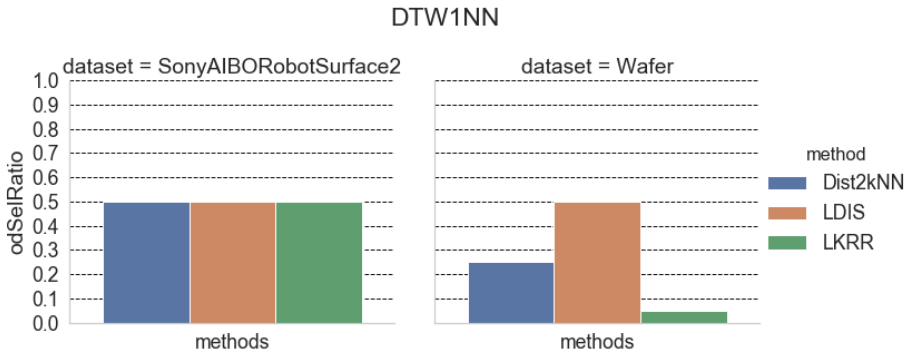


Figure 4.9.: OdSelRatio values for classifier DTW1NN and corresponding datasets.

4.4.3. Discussion of Results

It is important that this work should not be a comparison of the performance of different feature selection algorithms, nor a comparison of different classification algorithms. The focus of our experiments lies entirely on the impact of instance selection for feature selection on the classification performance. To investigate this impact we have to answer our main research question, divided into the following two questions:

- Are outlier detection algorithms applicable for filtering instances to influence feature selection algorithms?
- Do this adapted feature subsets have a positive impact on the performance of learning algorithms?

After analyzing the results of our experiments, we can answer both question with a clear "Yes". The results of our experiments suggest that for both classifiers the TP-rates, where we used some form of instance selection always increased, and therefore has a positive impact on the classification performance. Furthermore we found out that the used outlier detection algorithm (LKRR) is applicable for instance selection to positively influence the involved feature selection. We had no dataset where the the local baseline (classification only) was the best performing method, nor did we have a dataset where classification with attribute selection was better than the experiments involving instance selection. For a deeper analysis, we also want to investigate the following questions:

- What is the influence of the different datasets?
- What is the influence of different instance selection algorithms?
- What is the influence of the parameters `odSelRatios` and `odDescOrder`?
- What is the influence of parameters and components used for the nearest neighbor search: `kNeigbours`, `DistanceFunctions`?

The increase of performance of the classification strongly depends on the used datasets, components and parameters. We will inspect the specific questions in the following separate sections.

Influence of different Datasets

Concerning the performance of our approach given the different datasets, we found out that the results of our approach strongly depend on the used dataset. For datasets that already had an exceptional baseline result (for example ItalyPowerDemand and Wafer) the increase for the Rotation Forest classifier was only a matter of mills and only a matter of one or two percents for DTW1NN classifier. For these datasets we wanted to investigate whether the results might decrease, but this was not the case. Also the different classifiers had different increasing results given the different datasets. For example the DTW1NN classifier only had an increase of about two percent for the Lightning2 dataset, whereas the Rotation Forest classifier had an increase of eight percent. Interestingly both classifiers had the best performance for the SonyAIBORobotSurface1 dataset where DTW1NN increased for about 16 percent from the baselines and the Rotation Forest classifier increased ten percent. In general it seems that the lower the baseline performance of a dataset is, the higher is the increase of performance when performing instance selection for feature selection.

Influence of Instance Selection Algorithms

Referring to the different instance selection methods, we see that LKRR outlier detection algorithm had the largest increase in performance in thirteen out of sixteen experiment suites. We would like to draw the reader's attention to the fact that there are several equally performing methods for four experiments. Distance2KNN method only had the best performance for two out of sixteen experiment suites. Furthermore the instance selection method LDIS achieved the best performance for six out of sixteen experiment suites.

Influence of Instance Selection and Nearest Neighbor Search Parameters

This analysis is based on result parameter tables available in Section A.4 in the Appendix. For these tables we assembled for each experimental suite (recap: a combination of a dataset, a classifier and an instance selection

method) the best result and all of the used parameters. In the case that there was more than a best result for a suite, all the best results were added to the table.
We first start with the parameter of the instance selection methods, namely odSelRatio and odDescOrder. Given the Distance2kNN method the **odSelRatio** for both the Rotation Forest and DTW1NN classifier seems to have a weak dependency on the distance function for the Wafer and the Earthquakes datasets. For the other datasets it seems that the odSelRatio parameter depends on the dataset (so it seems there is one parameter value specific for a given dataset, where the performance is the best). This is the same for the LDIS method for the DTW1NN classifier and all datasets. For the Rotation Forest classifier this is only different for the ItalyPowerDemand, Lightning2 and Wafer datasets, were there are variations in the odSelRatio among the best performing experiment suits. For the LKRR method each best solution strictly depends only on one specific value for the odSelRatio regardless of the classifier used.
For the odSelRatio it is also relevant which values this parameter has for the best performing experiments (we also provided plots in Section 4.4). For the Rotation Forest classifier only the datasets Wafer and Lighting2 needed 90 percent of the instances for the Distance2kNN and the LKRR respectively. The other results indicate that for the best results in each experimental suite only between 25 and 50 percent of the original training instances are required.
This situation is different for the DTW1NN classifier. It seems that this classifier needed at least 90 percent of instances for some instance selection methods to achieve the best results. Although some datasets only required a smaller percentage of instances, for example for the Lighting2 and Wafer datasets the LDIS and the LKRR method only needed 5 percent of instances, which is interesting because this is the opposite picture shown for the Rotation Forest classifier.
For the parameter **odDescOrder** concerning the DTW1NN classifier and the methods LDIS and LKRR it seems that the parameter is strictly depended on the dataset. For the Rotation Forest classifier and LKRR method this is not true for the ItalyPowerDemand dataset and for the LDIS method not true for the same dataset and additionally the Lightning and Wafer dataset. Concerning the Distance2kNN method and both classifiers we where not able to find a distinctive pattern.

Furthermore, we would like to investigate the influence of the kNeighbors parameter and the DistanceFunctions component.
Given the Distance2kNN method the **kNeighbors** seems to be irrelevant, since we have best results for all three possible values. With regard to the LDIS method, the parameter is only independent for the datasets ItalyPowerDemand, Lightning, and Wafer, but for both classifiers. For the other datasets, there is a clear dependency on the dataset, again for both classifiers. For the LKRR method we have a different analysis for both classifiers. For the DTW1NN classifier kNeighbors seems to depend on the dataset, with the only exception for the Wafer dataset. For the Rotation Forest we have the same dependency on the dataset except for the ItalyPowerDemand and SonyAIBORobotSurface1 dataset.
Finally we investigate the influence of the **distance functions** concerning the results of our experiments. Given the Distance2kNN method all distance functions seem to depend on the dataset. For the LDIS method and the DTW1NN classifier we have independence given the datasets ItalyPowerDemand, MoteStrain and SonyAIBORobotSurface2. The same is true for the Rotation Forest classifier except also the Wafer dataset. For the other dataset it seems that the distance functions are dependent on the datasets. Given the LKRR method all datasets are dependent on the distance function except the ItalyPowerDemand and Wafer dataset for the DTW1NN classifier. For the Rotation Forest classifier this is also true except for the datasets ItalyPowerDemand and SonyAIBORobotSurface1.
In summary, for very high performance records (ItalyPowerDemand, Wafer), the likelihood that one parameter depends on the dataset is smaller. For most other datasets, it seems that the parameters are almost always dependent on the dataset.

5. Conclusions

The aim of this work was to overcome the problem of imbalanced datasets, where normal instances are over-represented, while rare events are typically very limited. To tackle this problem we applied outlier detection techniques as instance selection for these rare events, in order to influence following feature selection methods. Furthermore we showed that this approach lead to positive increases of the performance of trailing classifiers. Although outlier detection and instance selection are very similar topics, there have been very few attempts to combine these two techniques until this work. Additionally we found no evidence that instance selection was used only for feature selection methods before. With the help of a dedicated experiment pipeline, we were able to evaluate a very large number of experiments with a great deal of different components and parameters. These experiments were performed on a variety of different datasets using state of the art time series classification algorithms. The evaluation of this large number of experiments showed that the used outlier detection algorithm LocalKernelRidgeRegression outperformed a comparable instance selection algorithm. To closely inspect our approach, or to execute our experiments, we have made all used datasets, detailed results, and our entire source code available in a public git repository.

5.1. Outlook

Our future research focuses on the following topics:

- Especially for the DTW1NN classifier, the experiments for the larger datasets took a very long time ($>$ days). In this case, a parallelization

of these experiments would be desirable (for example starting experimental suits for certain parameters like attributeSelRatio in parallel)
- Due to the very large number of experiments, the manual analysis of the results is very time-consuming. Due to the large number of different parameters, we would like to automate the effect of the different components and parameters (for example, by means of correlation analyzes).
- Due to the actual improvement of the feature selection and the associated classification, the next step would be the impact of our approach on other wrappers or embedded methods.
- Since instance selection is a data reduction technique, experiments using random sampling would be an interesting comparison to our approach.
- These aforementioned sampling methods are already used in so-called ensemble methods. It would be interesting to examine the meaningfulness of our approach in this area.

Appendix

Appendix A.

Appendix

A.1. SOTA irrelevant results

Title	Year	Autors	Comment
A methodology for training set instance selection using mutual information in time series prediction	2014	Miloš B.Stojanovića, Miloš M.Božićb, Milena M.Stanković, Zoran P.Stajić	Pure IS paper
A survey on addressing high-class imbalance in big data	2018	Joffrey L. Leevy, Taghi M. Khoshgoftaar, Richard A. Bauder, Naeem Seliya	Wrong approach
A survey on data preprocessing for data stream mining: Current status and future directions	2017	Sergio Ramírez-Gallegoa, Bartosz Krawczyk, Salvador García, Michał Woźniak, FranciscoHerrera	Wrong approach
Bagging of Instance Selection Algorithms	2014	Marcin Blachnik, Mirosław Kordos	Pure IS paper
Data compression by volume prototypes for streaming data	2010	Kenji Tabata, Maiko Sato, Mineichi Kudo	Wrong approach
Enhancing the Quality of Noisy Training Data Using a Genetic Algorithm and Prototype Selection	2008	Boseon Byeon, Khaled Rasheed, Prashant Doshi	Wrong approach (IS before OD)
Ensemble Classifier for Mining Data Streams	2014	Ireneusz Czarnowski, Piotr Jedrzejowicz	Wrong approach
Ensemble learning for data stream analysis: A survey	2017	Bartosz Krawczyk, Leandro L.Minku, João Gama, Jerzy Stefanowski, Michał Woźniak	Wrong approach

Title	Year	Autors	Comment
Genetic algorithms in feature and instance selection	2013	Chih-Fong Tsai, William Eberle , Chi-Yuan Chu	Pure IS paper
Instance selection based on supervised clustering	2012	Jun-Hai Zhai, Hong-Yu Xui, Su-Fang Zhang, Na Li, Ta Li	Pure IS paper
Instance Selection for Classifier Performance Estimation in Meta Learning	2017	Marcin Blachnik	Wrong approach
k-Nearest neighbors optimization-based outlier removal	2015	Abraham Yosipof, Hanoch Senderowitz	Wrong approach
Learning to detect representative data for large scale instance selection	2015	Wei-Chao Lin, Chih-Fong Tsai, Shih-Wen Ke, Chia-Wen Hung, William Eberle	Wrong approach
Manifolds for training set selection through outlier detection	2011	Ahmed S. Tolba	Wrong approach
Multiple-instance learning with pairwise instance similarity	2014	Liming Yuan, Jiafeng Liu, Xianglong Tang	Wrong approach
Mutual Information-Based Input Selection for Electric Load Time Series Forecasting	2013	Miloš Božić, Miloš Stojanović, Zoran Stajić, Nenad Floranović	Wrong approach
New method for instance or prototype selection using mutual information in time series prediction	2010	A.Guillen, L.J.Herrera, G.Rubio, H.Pomares, A.Lendasse, I.Rojas	Pure IS paper

Title	Year	Autors	Comment
Noisy data elimination using mutual k-nearest neighbor for classification mining	2012	Huawen Liu, Huawen Liu	Wrong approach
Outlier detection for training-based adaptive protocols	2013	Hui Liu, Jialin He, Dinesh Rajan, Joseph Camp	Wrong field (adaptive protocols)
Parallel MCNN (pMCNN) with Application to Prototype Selection on Large and Streaming Dat	2017	V. Susheela Devi, Lakhpat Meena	Wrong approach
Recent advances in scaling-down sampling methods in machine learning	2017	Amr ElRafey, Janusz Wojtusiak	Wrong approach
Training set selection for the prediction of essential genes	2014	Cheng J, Xu Z, Wu W, Zhao L, Li X, Liu Y, Tao S	Pure IS paper

Table A.1.: Results of the SOTA analysis of outlier detection for instance selection which turned out to be not relevant.

A.2. Extended performance metrics

A.2.1. Extended performance metrics Rotation Forest

Dataset	ExpSuite	Method	TP_RATE	FP_RATE	Precision	Recall	F1
Earthquakes	LocBase	-	0.755	0.613	0.718	0.755	0.711
Earthquakes	AS-Only	SymmUncert	0.806	0.520	0.802	0.806	0.773
Earthquakes		LKRR	0.813	0.518	0.819	0.813	0.779
FordA	LocBase	-	0.749	0.252	0.749	0.749	0.749
FordA	AS-Only	InfoGain SymmUncert GainRatio	0.760	0.241	0.760	0.760	0.760
FordA	With IS	LKRR	0.789	0.211	0.789	0.789	0.789
ItalyPower Demand	LocBase	-	0.969	0.031	0.969	0.969	0.969
ItalyPower Demand	AS-Only	OneR	0.974	0.026	0.974	0.974	0.974
ItalyPower Demand	With IS	LKRR	0.976	0.024	0.976	0.976	0.976
Lighting2	LocBase	-	0.754	0.247	0.755	0.754	0.754
Lighting2	AS-Only	GainRatio	0.787	0.213	0.788	0.787	0.787
Lighting2	With IS	Dist2KNN LKRR	0.836	0.166	0.836	0.836	0.836
MoteStrain	LocBase	-	0.868	0.135	0.868	0.868	0.868
MoteStrain	AS-Only	InfoGain	0.887	0.119	0.887	0.887	0.886
MoteStrain	With IS	LKRR	0.906	0.093	0.907	0.906	0.906
SonyAIBORobot Surface1	LocBase	-	0.775	0.172	0.846	0.775	0.772
SonyAIBORobot Surface1	AS-Only	InfoGain	0.799	0.153	0.859	0.799	0.797

Dataset	ExpSuite	Method	TP_RATE	FP_RATE	Precision	Recall	F1
SonyAIBORobot Surface1	With IS	LDIS	0.885	0.100	0.895	0.885	0.886
SonyAIBORobot Surface2	LocBase	-	0.779	0.227	0.785	0.779	0.780
SonyAIBORobot Surface2	AS-Only	Relief	0.837	0.216	0.839	0.837	0.834
SonyAIBORobot Surface2	With IS	LKRR	0.856	0.188	0.857	0.856	0.854
Wafer	LocBase	-	0.992	0.047	0.992	0.992	0.992
Wafer	AS-Only	InfoGain	0.997	0.015	0.997	0.997	0.997
Wafer	With IS	LKRR	1.000	0.004	1.000	1.000	1.000

Table A.2.: Extended version of performance metrics for Rotation Forest classifier. Shown are only the local baseline, best attribute selector and best instance selection for each dataset.

A.2.2. Extended performance metrics DTW1NN

Dataset	ExpSuite	Method	TP_RATE	FP_RATE	Precision	Recall	F1
Earthquakes	LocBase	-	0.719	0.644	0.666	0.719	0.679
Earthquakes	AS-Only	SymmUncert	0.755	0.518	0.731	0.755	0.737
Earthquakes	With IS	LKRR	0.799	0.485	0.783	0.799	0.776
FordA	LocBase	-	0.555	0.449	0.554	0.555	0.553
FordA	AS-Only	OneR	0.626	0.381	0.628	0.626	0.622
FordA	With IS	LKRR	0.655	0.349	0.655	0.655	0.654
ItalyPower Demand	LocBase	-	0.950	0.049	0.951	0.950	0.950
ItalyPower Demand	AS-Only	OneR	0.976	0.024	0.976	0.976	0.976
ItalyPower Demand	With IS	LKRR	0.977	0.023	0.977	0.977	0.977
Lighting2	LocBase	-	0.869	0.149	0.882	0.869	0.866
Lighting2	AS-Only	SymmUncert	0.820	0.185	0.820	0.820	0.819
Lighting2	With IS	LDIS	0.885	0.119	0.885	0.885	0.885
MoteStrain	LocBase	-	0.835	0.167	0.835	0.835	0.835
MoteStrain	AS-Only	OneR	0.879	0.124	0.879	0.879	0.879
MoteStrain	With IS	Dist2KNN LDIS LKRR	0.890	0.114	0.890	0.890	0.890
SonyAIBORobot Surface1	LocBase	-	0.725	0.207	0.830	0.725	0.716
SonyAIBORobot Surface1	AS-Only	InfoGain	0.799	0.151	0.863	0.799	0.796

Dataset	ExpSuite	Method	TP_RATE	FP_RATE	Precision	Recall	F1
SonyAIBORobot Surface1	With IS	LDIS	0.885	0.090	0.904	0.885	0.886
SonyAIBORobot Surface1	LocBase	-	0.831	0.210	0.830	0.831	0.829
SonyAIBORobot Surface2	AS-Only	Relief	0.850	0.176	0.849	0.850	0.849
SonyAIBORobot Surface2	With IS	LKRR	0.886	0.142	0.885	0.886	0.885
Wafer	LocBase	-	0.980	0.145	0.980	0.980	0.979
Wafer	AS-Only	Relief	0.991	0.039	0.991	0.991	0.991
Wafer	With IS	LDIS LKRR	0.995	0.036	0.995	0.995	0.995

Table A.3.: Extended version of performance metrics for DTW1NN classifier. Shown are only the local baseline, best attribute selector and best instance selection for each dataset.

A.3. Result plots

A.3.1. Result plots Rotation Forest

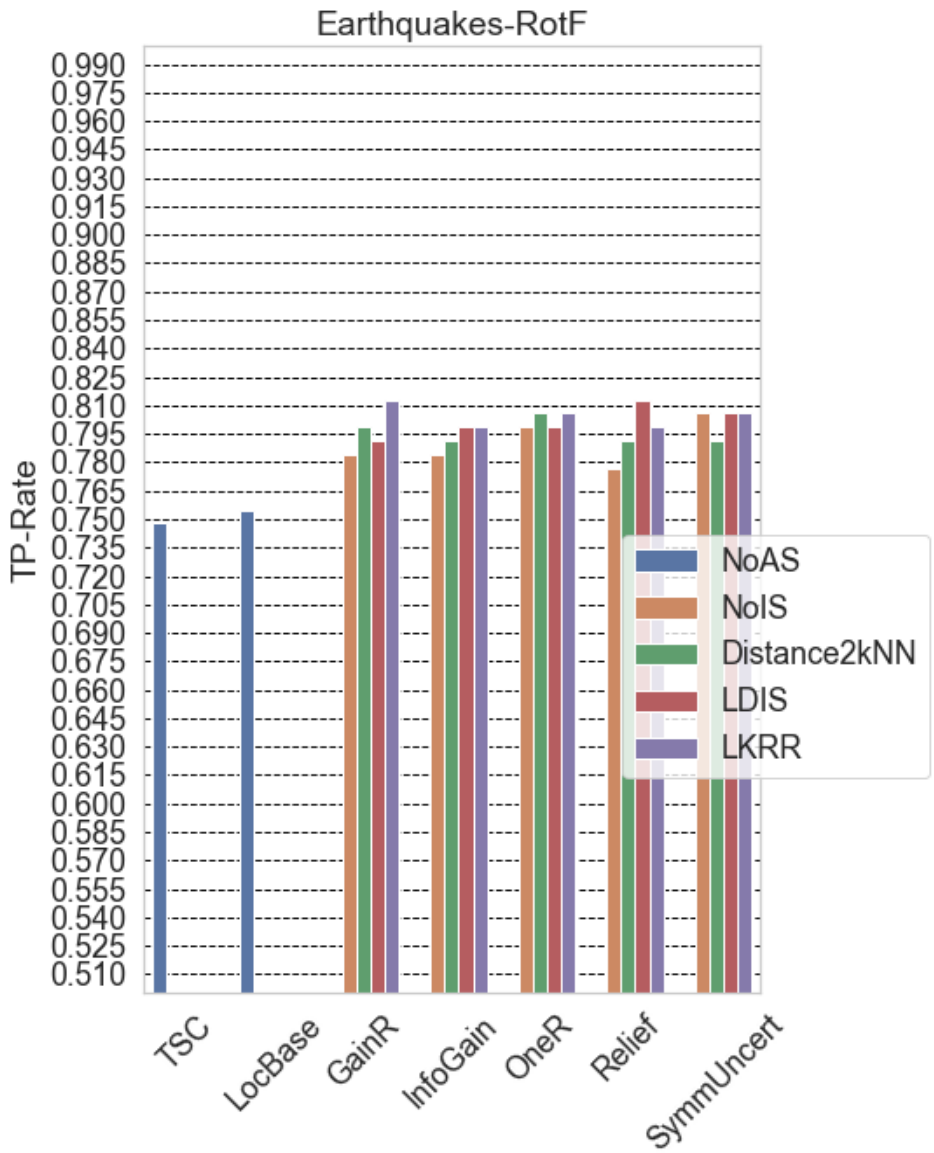


Figure A.1.: Results for dataset Earthquakes for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier Rotation Forest.

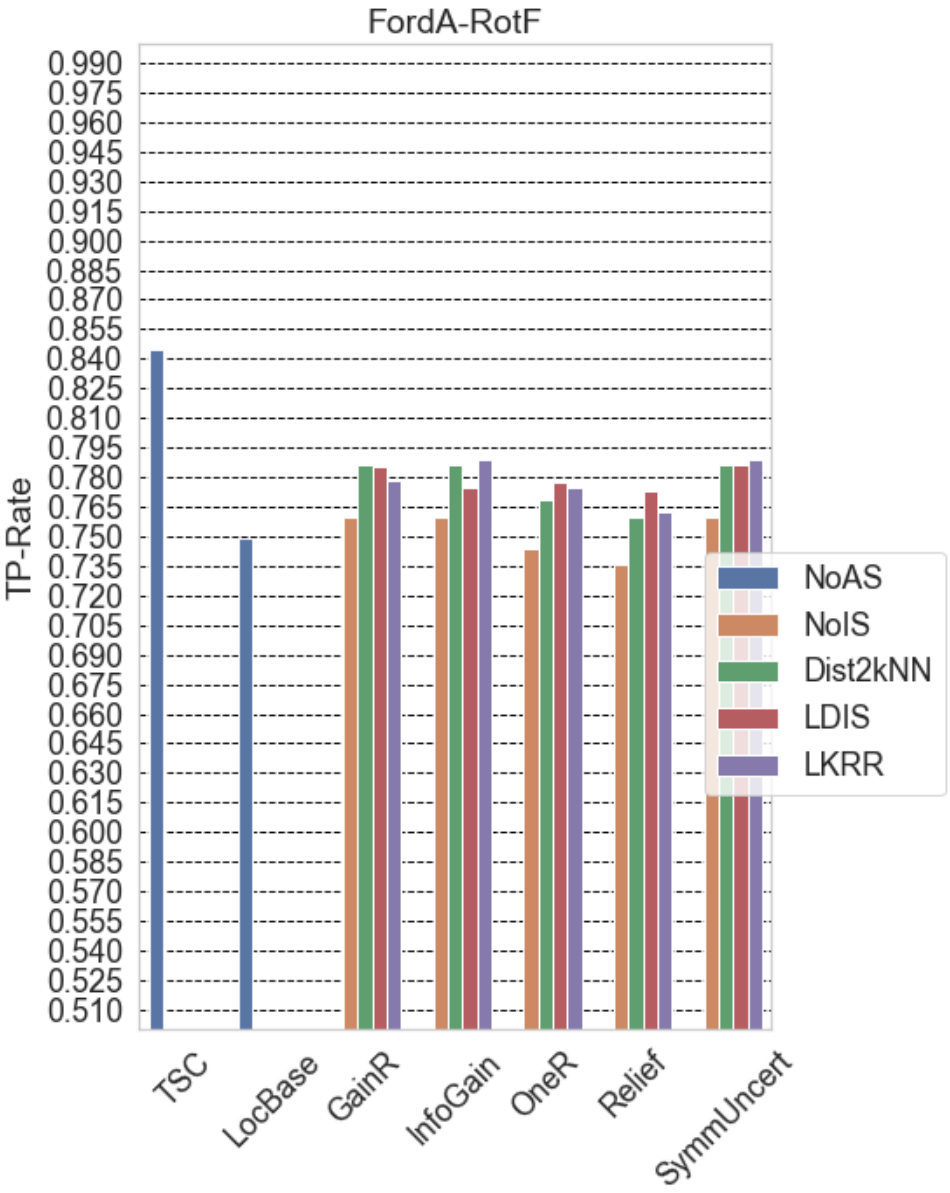


Figure A.2.: Results for dataset FordA for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier Rotation Forest.

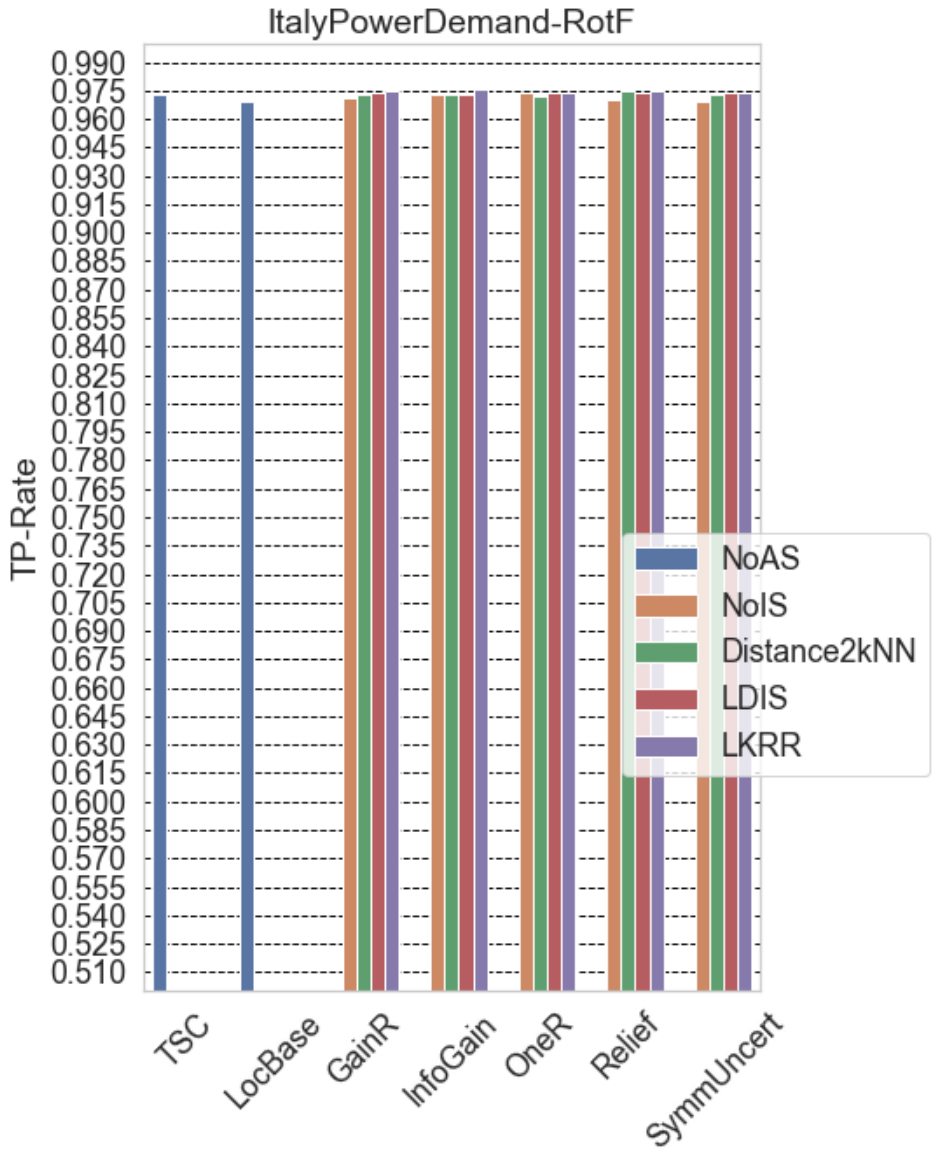


Figure A.3.: Results for dataset ItalyPowerDemand for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier Rotation Forest.

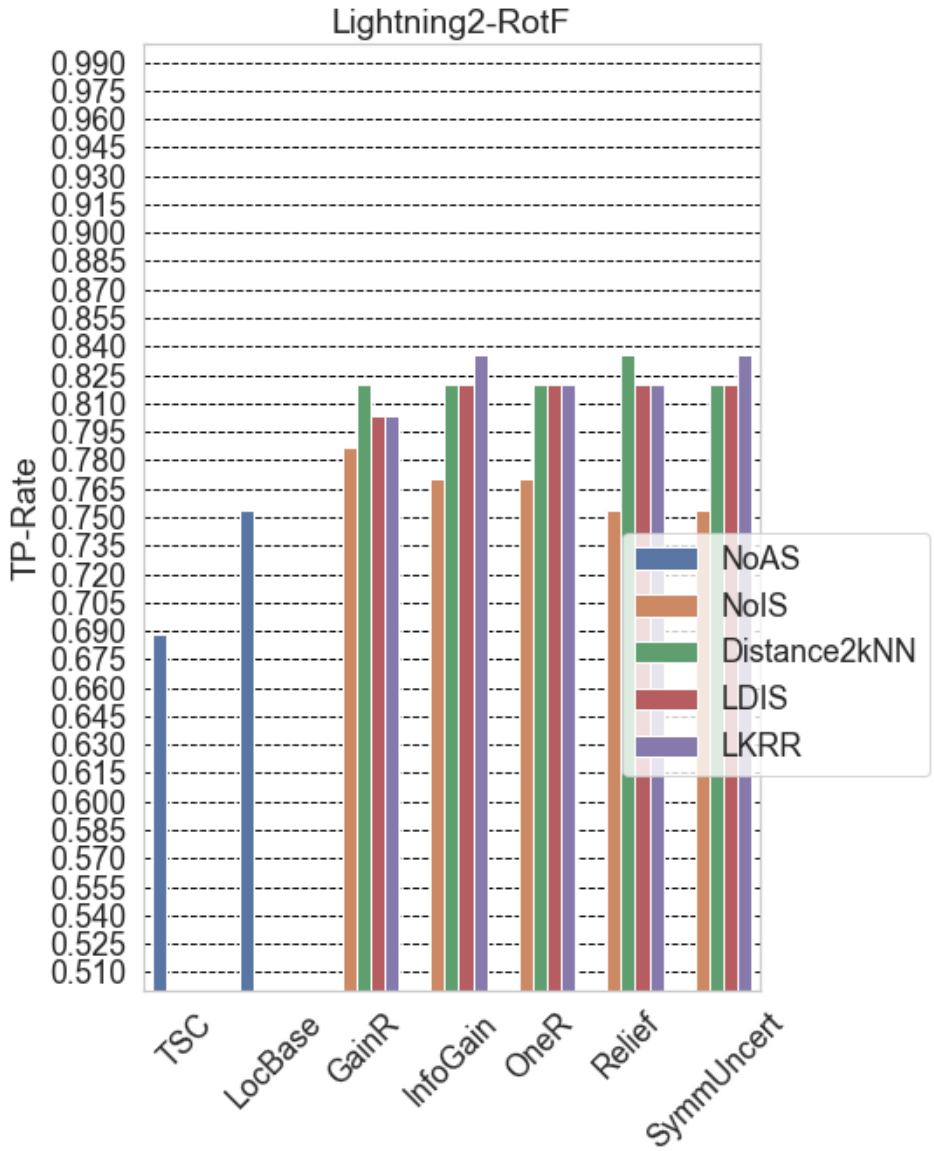


Figure A.4.: Results for dataset Lightning2 for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier Rotation Forest.

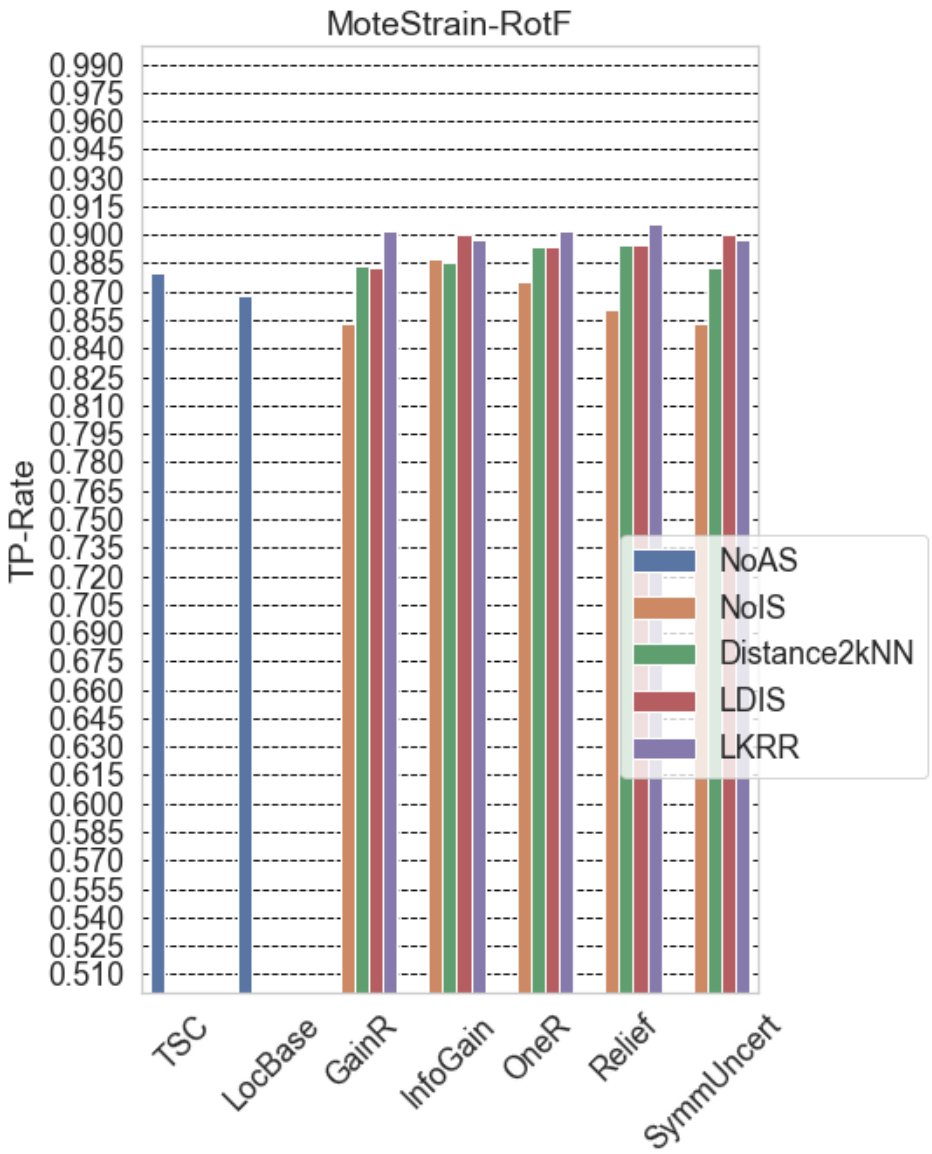


Figure A.5.: Results for dataset MoteStrain for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier Rotation Forest.

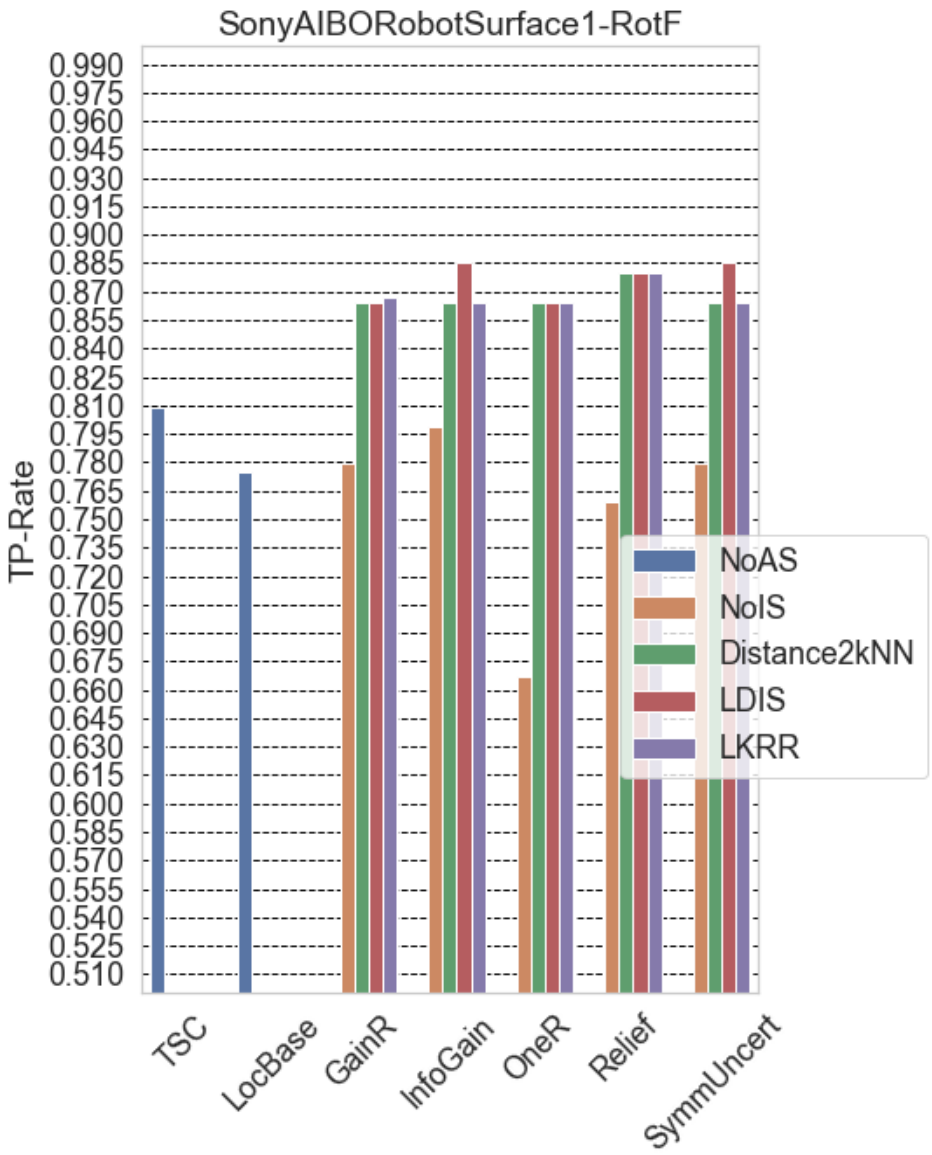


Figure A.6.: Results for dataset SonyAIBORobotSurface1 for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier Rotation Forest.

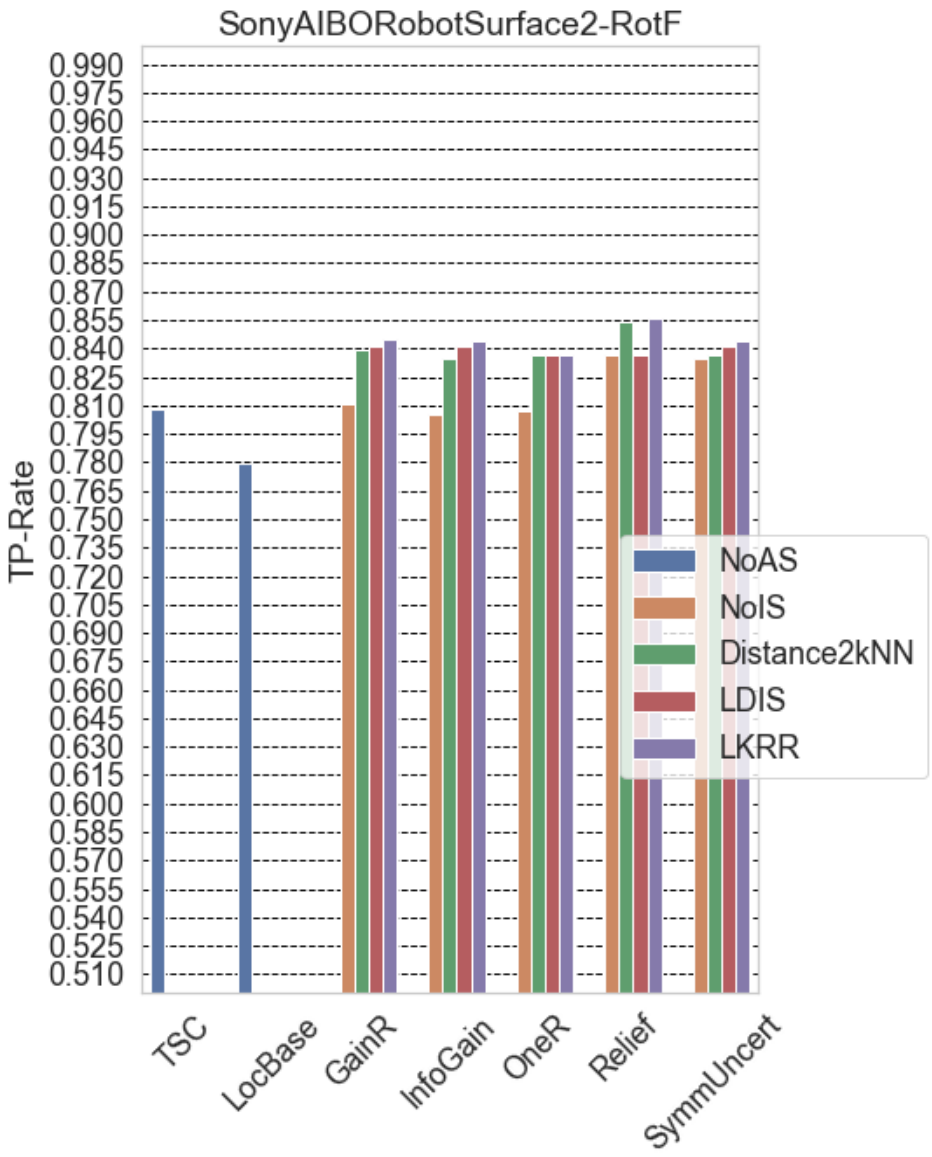


Figure A.7.: Results for dataset SonyAIBORobotSurface2 for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier Rotation Forest.

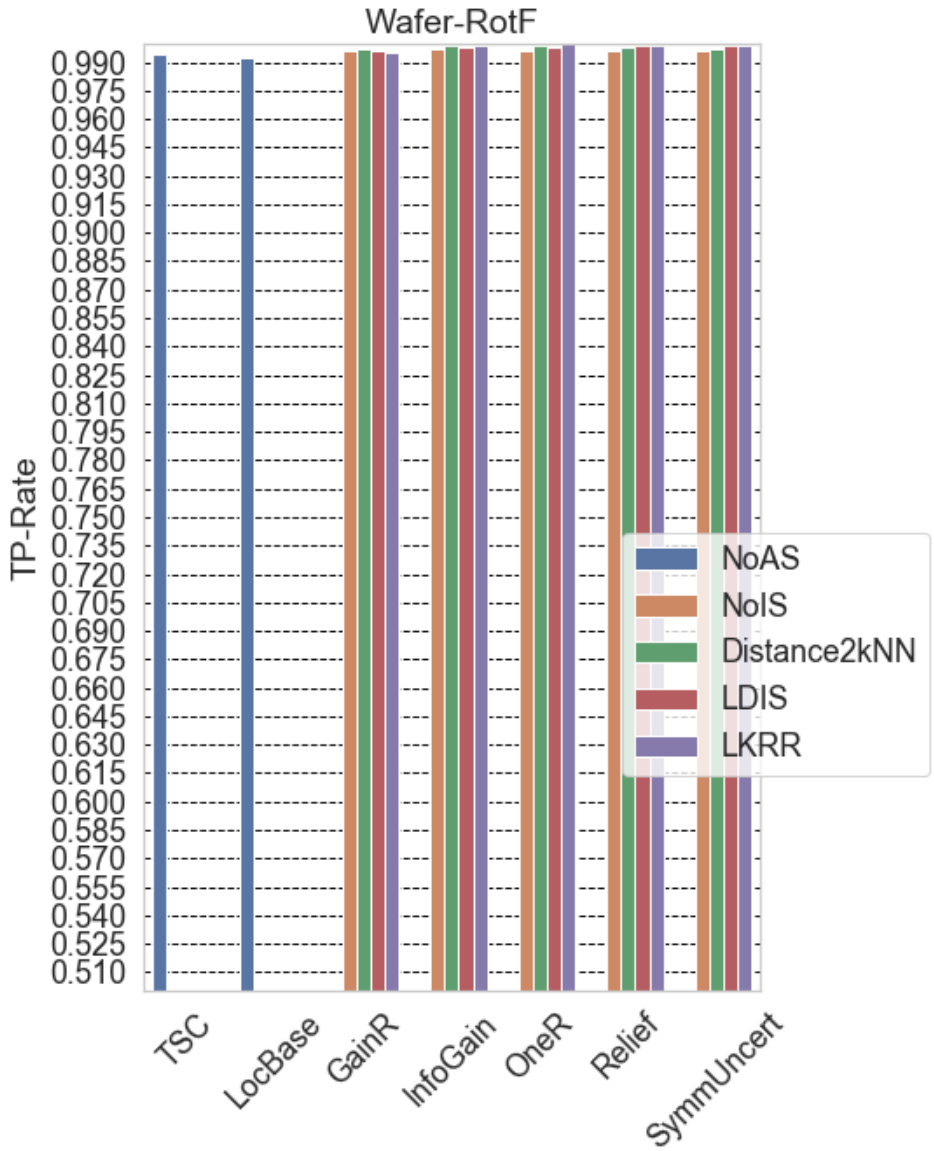


Figure A.8.: Results for dataset Wafer for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier Rotation Forest.

A.3.2. Results plots DTW1NN

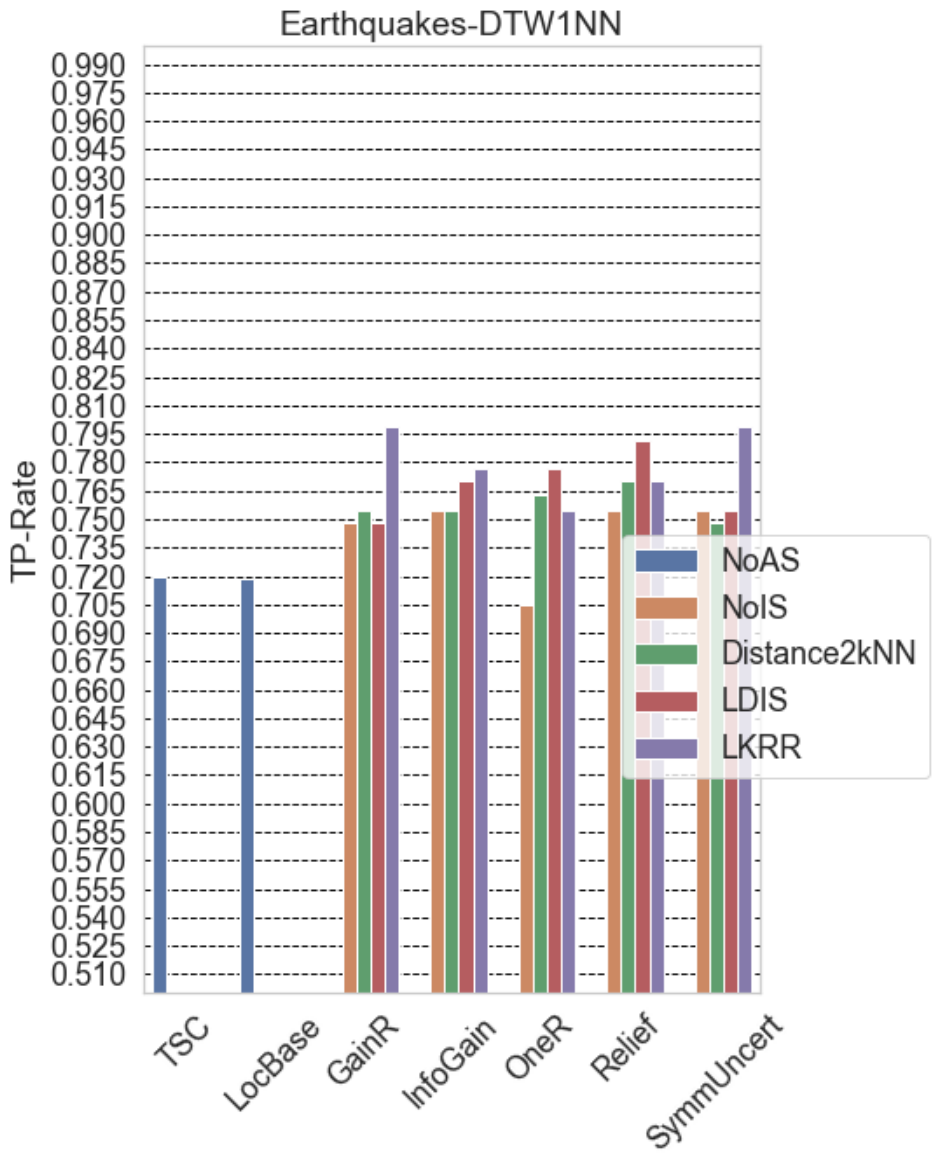


Figure A.9.: Results for dataset Earthquakes for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier DTW1NN.

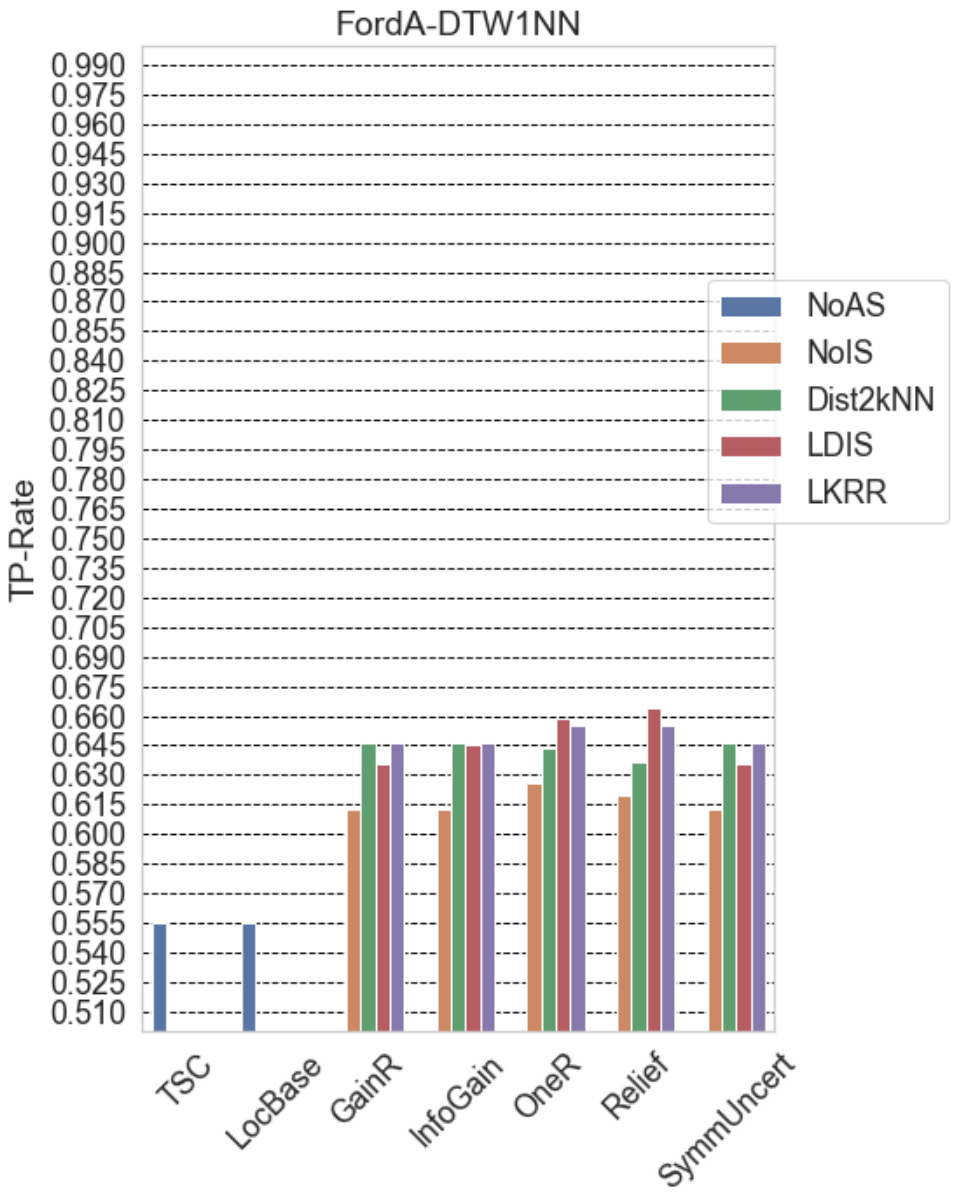


Figure A.10.: Results for dataset FordA for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier DTW1NN.

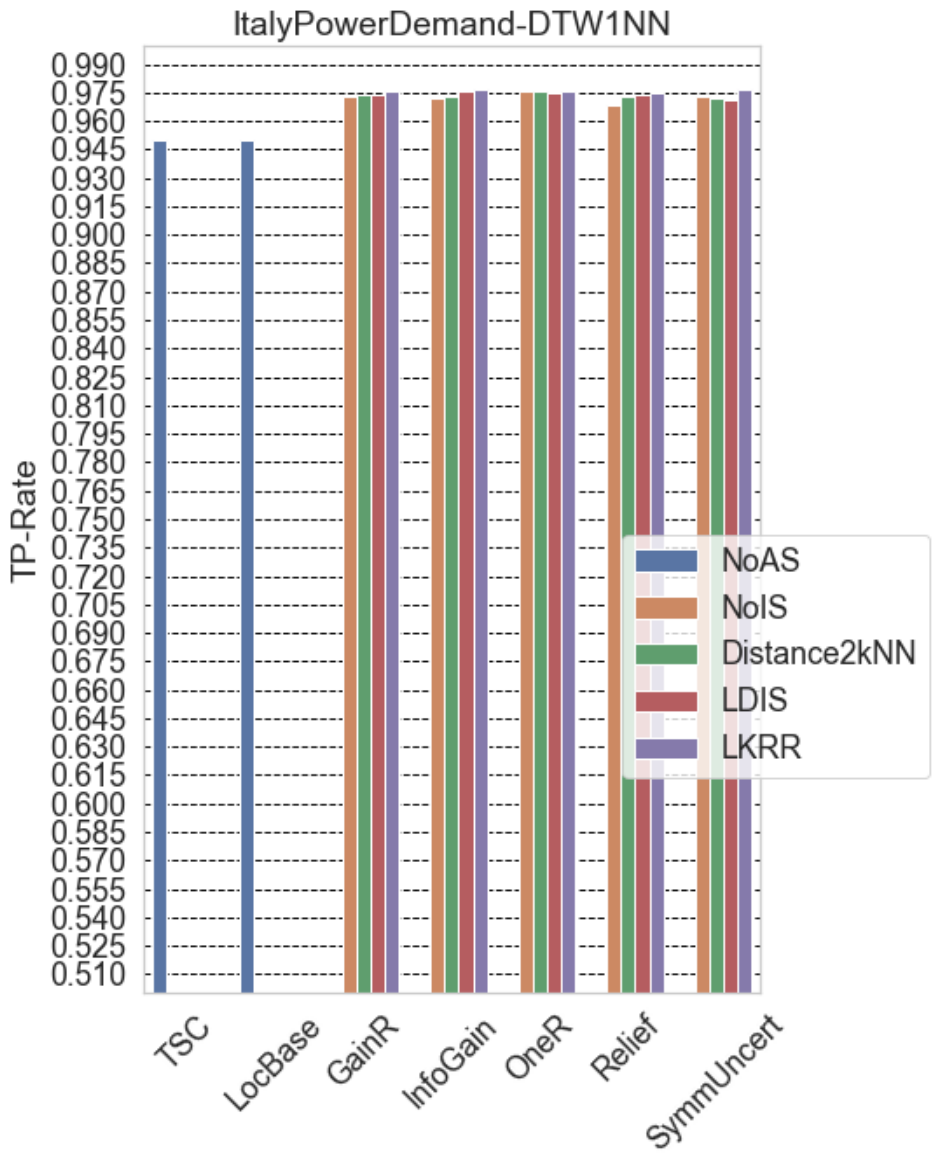


Figure A.11.: Results for dataset ItalyPowerDemand for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier DTW1NN.

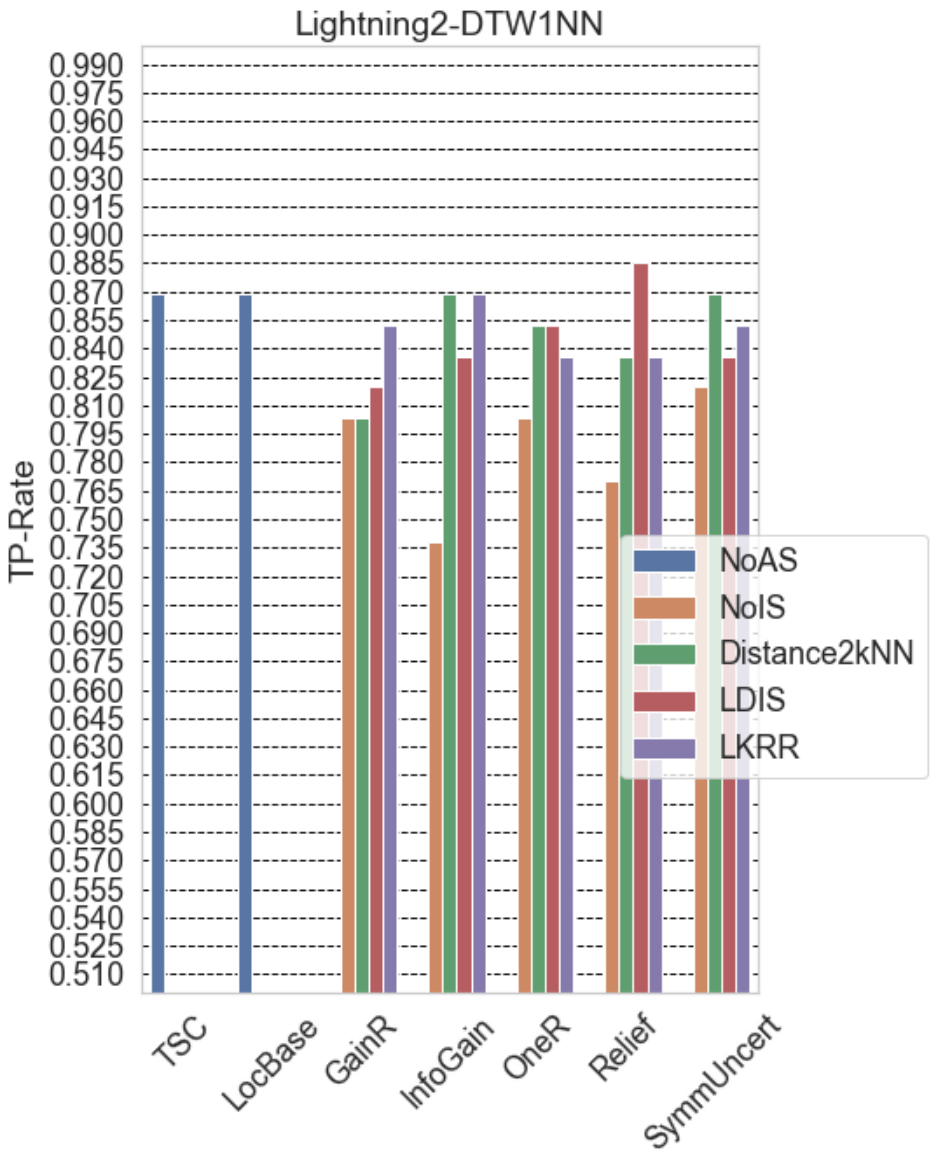


Figure A.12.: Results for dataset Lightning2 for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier DTW1NN.

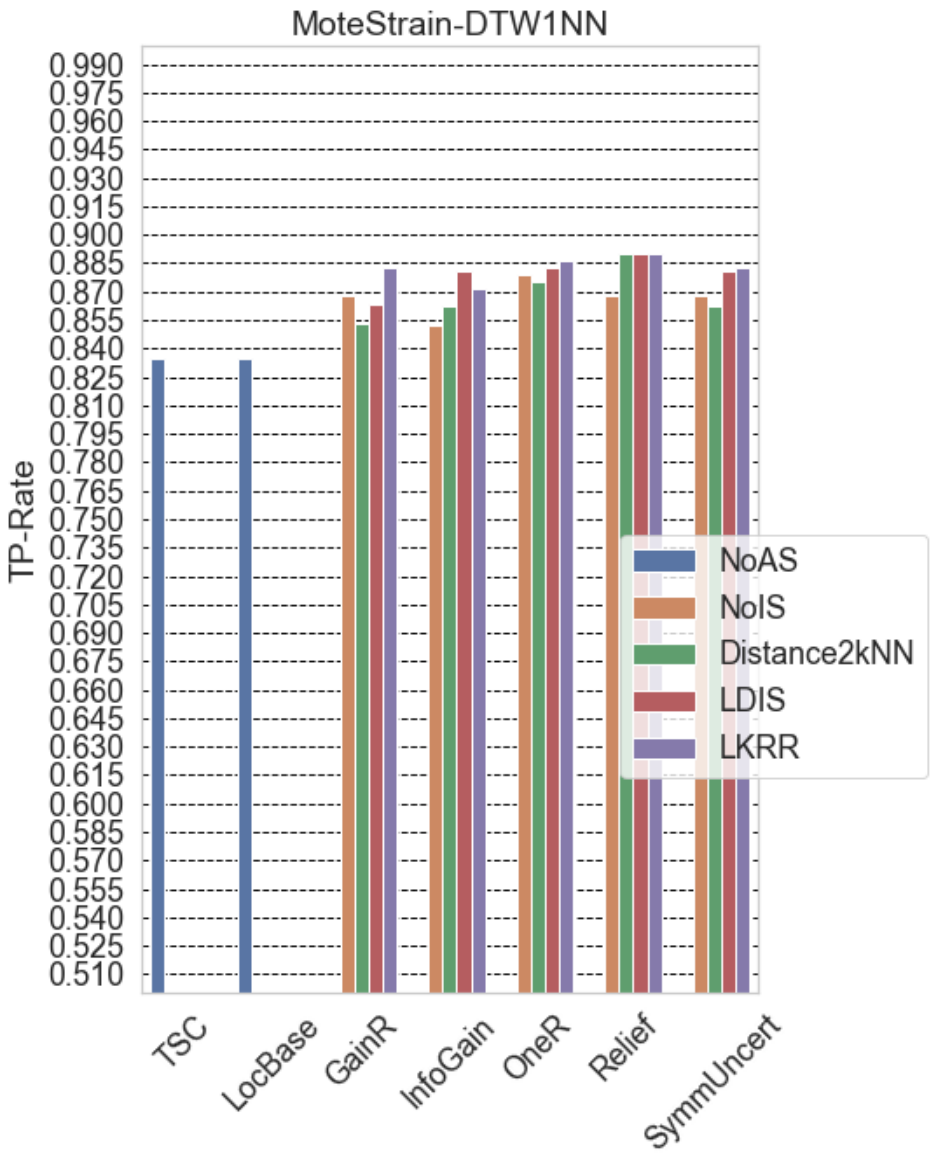


Figure A.13.: Results for dataset MoteStrain for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier DTW1NN.

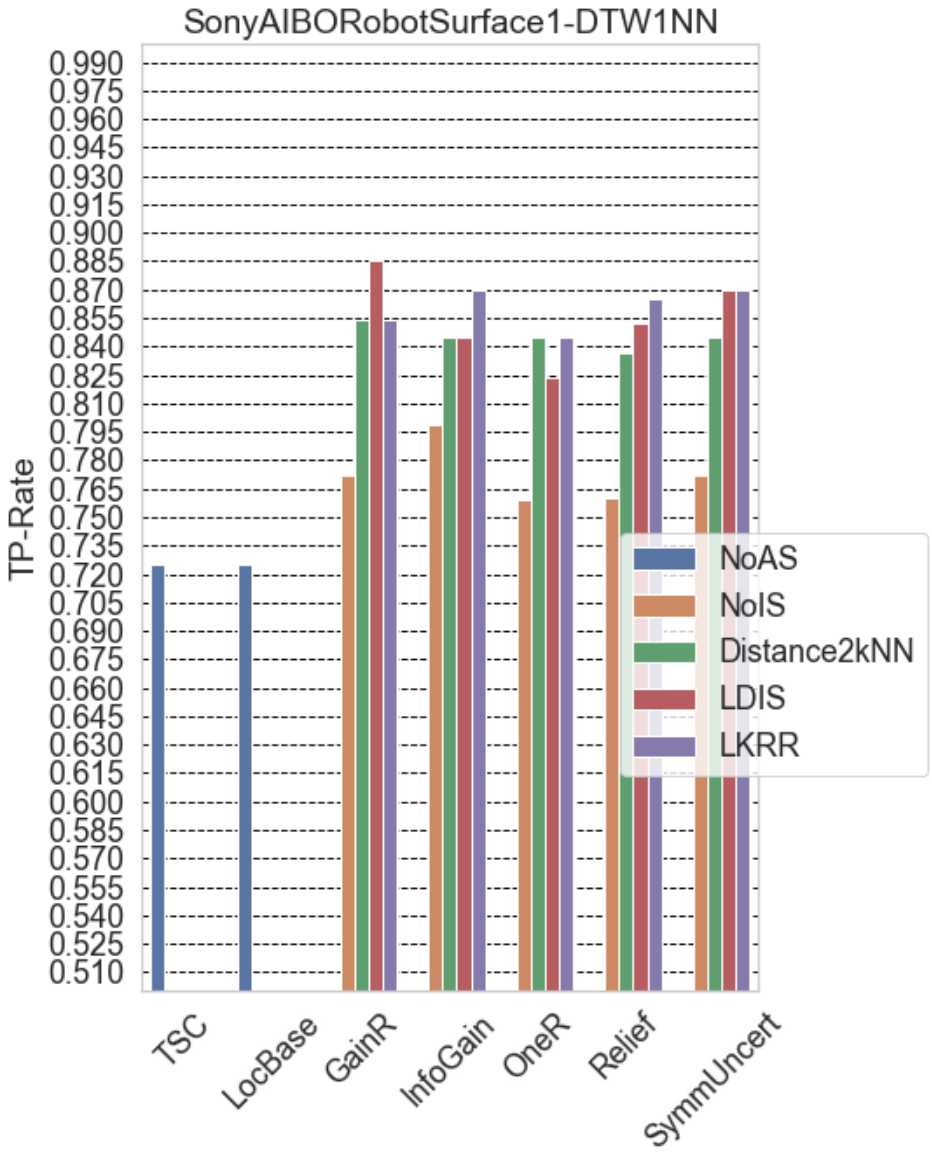


Figure A.14.: Results for dataset SonyAIBORobotSurface1 for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier DTW1NN.

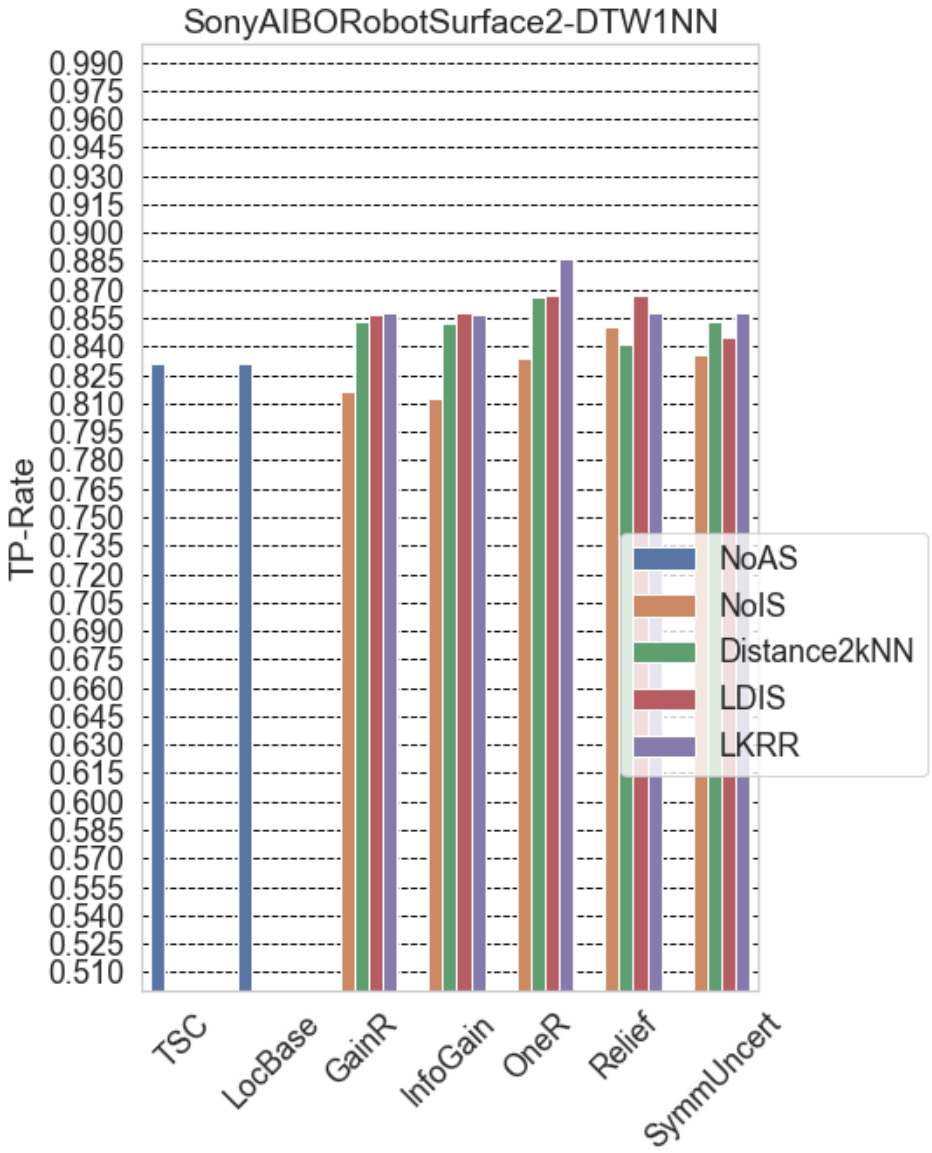


Figure A.15.: Results for dataset SonyAIBORobotSurface2 for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier DTW1NN.

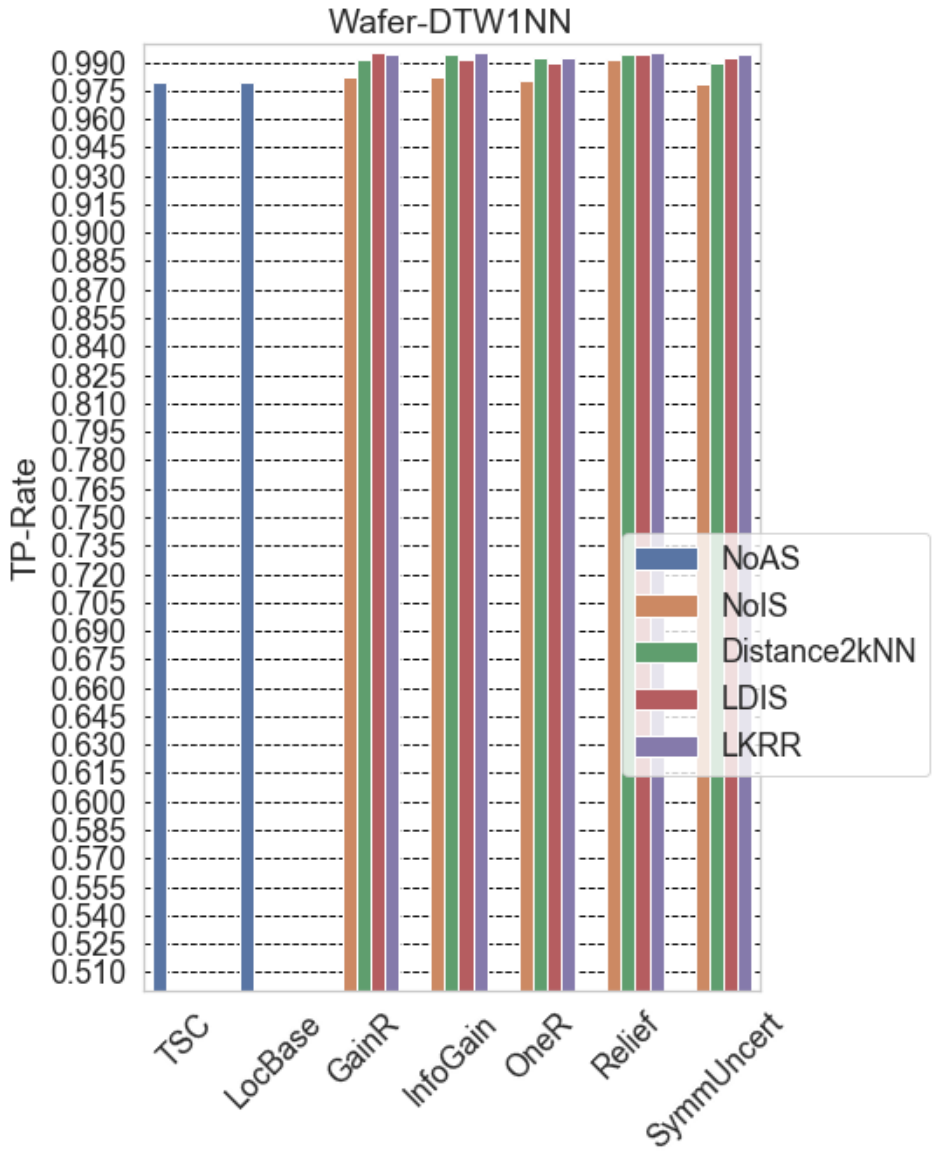


Figure A.16.: Results for dataset Wafer for all methods (including baseline from TSC and local baseline) and attribute selectors for classifier DTW1NN.

A.4. Result parameter tables

A.4.1. Result parameter tables Rotation Forest

dataset	kNeighbors	odSelRatio	odDistFunc	odDescOrder	TP_RATE
Earthquakes	1	0.25	Euclidean	false	0.806
Earthquakes	1	0.25	Minkowski	false	0.806
Earthquakes	5	0.25	Euclidean	false	0.806
Earthquakes	5	0.25	Minkowski	false	0.806
Earthquakes	10	0.25	Euclidean	false	0.806
Earthquakes	10	0.25	Minkowski	false	0.806
Earthquakes	1	0.05	Manhattan	true	0.806
Earthquakes	5	0.05	Manhattan	true	0.806
Earthquakes	10	0.05	Manhattan	true	0.806
FordA	1	0.25	Euclidean	true	0.786
FordA	1	0.25	Minkowski	true	0.786
FordA	5	0.25	Euclidean	true	0.786
FordA	5	0.25	Minkowski	true	0.786
FordA	10	0.25	Euclidean	true	0.786
FordA	10	0.25	Minkowski	true	0.786
ItalyPower Demand	1	0.5	Manhattan	false	0.975
ItalyPower Demand	5	0.5	Manhattan	false	0.975
ItalyPower Demand	10	0.5	Manhattan	false	0.975
ItalyPower Demand	1	0.9	Manhattan	true	0.975
ItalyPower Demand	5	0.9	Manhattan	true	0.975

dataset	kNeighbors	odSelRatio	odDistFunc	odDescOrder	TP_RATE
ItalyPower Demand	10	0.9	Manhattan	true	0.975
Lighting2	1	0.9	Euclidean	true	0.836
Lighting2	1	0.9	Minkowski	true	0.836
Lighting2	5	0.9	Euclidean	true	0.836
Lighting2	5	0.9	Minkowski	true	0.836
Lighting2	10	0.9	Euclidean	true	0.836
Lighting2	10	0.9	Minkowski	true	0.836
MoteStrain	1	0.5	Chebyshev	false	0.895
MoteStrain	5	0.5	Chebyshev	false	0.895
MoteStrain	10	0.5	Chebyshev	false	0.895
MoteStrain	1	0.5	Chebyshev	false	0.894
MoteStrain	5	0.5	Chebyshev	false	0.894
MoteStrain	10	0.5	Chebyshev	false	0.894
SonyAIBORobot Surface1	1	0.25	Chebyshev	false	0.88
SonyAIBORobot Surface1	5	0.25	Chebyshev	false	0.88
SonyAIBORobot Surface1	10	0.25	Chebyshev	false	0.88
SonyAIBORobot Surface2	1	0.5	Manhattan	false	0.854
SonyAIBORobot Surface2	5	0.5	Manhattan	false	0.854

dataset	kNeighbors	odSelRatio	odDistFunc	odDescOrder	TP_RATE
SonyAIBORobot Surface2	10	0.5	Manhattan	false	0.854
Wafer	1	0.25	Euclidean	false	0.999
Wafer	1	0.25	Minkowski	false	0.999
Wafer	5	0.25	Euclidean	false	0.999
Wafer	5	0.25	Minkowski	false	0.999
Wafer	10	0.25	Euclidean	false	0.999
Wafer	10	0.25	Minkowski	false	0.999
Wafer	1	0.5	Manhattan	true	0.999
Wafer	5	0.5	Manhattan	true	0.999
Wafer	10	0.5	Manhattan	true	0.999

Table A.4.: Result parameters for classifier RotationForest and instance selection method Distance2KNN

dataset	kNeighbors	odSelRatio	odDistFunc	odDescOrder	TP_RATE
Earthquakes	10	0.5	Chebyshev	true	0.799
FordA	10	0.25	Chebyshev	true	0.786
ItalyPower	5	0.05	Manhattan	false	0.974
Demand	5	0.05	Manhattan	false	0.974
ItalyPowerDemand	10	0.05	Euclidean	false	0.974
ItalyPowerDemand	10	0.05	Manhattan	false	0.974
ItalyPowerDemand	10	0.05	Minkowski	false	0.974
ItalyPowerDemand	1	0.5	Manhattan	false	0.974
ItalyPowerDemand	1	0.9	Chebyshev	false	0.974
ItalyPowerDemand	5	0.5	Chebyshev	false	0.974
ItalyPowerDemand	5	0.9	Chebyshev	true	0.974
ItalyPowerDemand	10	0.9	Manhattan	false	0.974
Lighting2	5	0.5	Manhattan	false	0.82
Lighting2	5	0.9	Manhattan	false	0.82
Lighting2	10	0.9	Manhattan	true	0.82
MoteStrain	5	0.5	Euclidean	true	0.9
MoteStrain	5	0.5	Euclidean	true	0.9
MoteStrain	5	0.5	Minkowski	true	0.9
MoteStrain	5	0.5	Minkowski	true	0.9
SonyAIBORobotSurface1	10	0.5	Manhattan	false	0.885
SonyAIBORobotSurface1	10	0.5	Manhattan	false	0.885
SonyAIBORobotSurface2	1	0.25	Euclidean	false	0.841
SonyAIBORobotSurface2	1	0.25	Manhattan	false	0.841
SonyAIBORobotSurface2	1	0.25	Manhattan	false	0.841
SonyAIBORobotSurface2	1	0.25	Minkowski	false	0.841

dataset	kNeighbors	odSelRatio	odDistFunc	odDescOrder	TP_RATE
Wafer	1	0.5	Chebyshev	true	0.999
Wafer	1	0.5	Euclidean	false	0.999
Wafer	1	0.5	Minkowski	false	0.999
Wafer	5	0.5	Chebyshev	true	0.999
Wafer	10	0.5	Chebyshev	true	0.999
Wafer	10	0.9	Euclidean	false	0.999
Wafer	10	0.9	Minkowski	false	0.999
Wafer	10	0.5	Manhattan	true	0.999

Table A.5.: Result parameters for classifier RotationForest and instance selection method LDIS

dataset	kNeighbors	odSelRatio	odDistFunc	odDescOrder	TP_RATE
Earthquakes	5	0.25	Manhattan	false	0.813
FordA	1	0.25	Manhattan	false	0.789
ItalyPowerDemand	5	0.25	Manhattan	true	0.976
ItalyPowerDemand	1	0.5	Euclidean	false	0.976
ItalyPowerDemand	1	0.5	Minkowski	false	0.976
Lighting2	1	0.25	Euclidean	false	0.836
Lighting2	1	0.25	Euclidean	false	0.836
Lighting2	1	0.25	Minkowski	false	0.836
Lighting2	1	0.25	Minkowski	false	0.836
MoteStrain	5	0.5	Manhattan	false	0.906
SonyAIBORobotSurface1	1	0.25	Chebyshev	false	0.88
SonyAIBORobotSurface1	5	0.25	Chebyshev	false	0.88
SonyAIBORobotSurface1	10	0.25	Chebyshev	false	0.88
SonyAIBORobotSurface2	10	0.5	Euclidean	false	0.856
SonyAIBORobotSurface2	10	0.5	Manhattan	false	0.856
SonyAIBORobotSurface2	10	0.5	Minkowski	false	0.856
Wafer	5	0.9	Manhattan	false	1

Table A.6.: Result parameters for classifier RotationForest and instance selection method LKRR

A.4.2. Result parameter tables DTW1NN

dataset	kNeighbors	odSelRatio	odDistFunc	odDescOrder	TP_RATE
Earthquakes	1	0.25	Euclidean	true	0.755
Earthquakes	1	0.25	Minkowski	true	0.755
Earthquakes	5	0.25	Euclidean	true	0.755
Earthquakes	5	0.25	Minkowski	true	0.755
Earthquakes	10	0.25	Euclidean	true	0.755
Earthquakes	10	0.25	Minkowski	true	0.755
FordA	5	0.25	Minkowski	false	0.646
FordA	5	0.25	Minkowski	false	0.646
FordA	5	0.25	Minkowski	false	0.646
FordA	1	0.25	Euclidean	false	0.646
FordA	1	0.25	Euclidean	false	0.646
FordA	1	0.25	Euclidean	false	0.646
FordA	10	0.25	Euclidean	false	0.646
FordA	10	0.25	Euclidean	false	0.646
FordA	10	0.25	Euclidean	false	0.646
FordA	1	0.25	Minkowski	false	0.646
FordA	1	0.25	Minkowski	false	0.646
FordA	1	0.25	Minkowski	false	0.646
FordA	5	0.25	Euclidean	false	0.646
FordA	5	0.25	Euclidean	false	0.646
FordA	5	0.25	Euclidean	false	0.646
FordA	10	0.25	Minkowski	false	0.646
FordA	10	0.25	Minkowski	false	0.646
FordA	10	0.25	Minkowski	false	0.646
ItalyPowerDemand	1	0.9	Euclidean	false	0.976

dataset	kNeighbors	odSelRatio	odDistFunc	odDescOrder	TP_RATE
ItalyPowerDemand	1	0.9	Manhattan	false	0.976
ItalyPowerDemand	1	0.9	Minkowski	false	0.976
ItalyPowerDemand	5	0.9	Euclidean	false	0.976
ItalyPowerDemand	5	0.9	Manhattan	false	0.976
ItalyPowerDemand	5	0.9	Minkowski	false	0.976
ItalyPowerDemand	10	0.9	Euclidean	false	0.976
ItalyPowerDemand	10	0.9	Manhattan	false	0.976
ItalyPowerDemand	10	0.9	Minkowski	false	0.976
Lighting2	1	0.25	Euclidean	true	0.869
Lighting2	1	0.25	Euclidean	true	0.869
Lighting2	1	0.25	Minkowski	true	0.869
Lighting2	1	0.25	Minkowski	true	0.869
Lighting2	5	0.25	Euclidean	true	0.869
Lighting2	5	0.25	Euclidean	true	0.869
Lighting2	5	0.25	Minkowski	true	0.869
Lighting2	5	0.25	Minkowski	true	0.869
Lighting2	10	0.25	Euclidean	true	0.869
Lighting2	10	0.25	Euclidean	true	0.869
Lighting2	10	0.25	Minkowski	true	0.869
Lighting2	10	0.25	Minkowski	true	0.869
MoteStrain	1	0.9	Euclidean	false	0.89
MoteStrain	1	0.9	Manhattan	false	0.89
MoteStrain	1	0.9	Minkowski	false	0.89
MoteStrain	5	0.9	Euclidean	false	0.89
MoteStrain	5	0.9	Manhattan	false	0.89

dataset	kNeighbors	odSelRatio	odDistFunc	odDescOrder	TP_RATE
MoteStrain	5	0.9	Minkowski	false	0.89
MoteStrain	10	0.9	Euclidean	false	0.89
MoteStrain	10	0.9	Manhattan	false	0.89
MoteStrain	10	0.9	Minkowski	false	0.89
SonyAIBORobotSurface1	1	0.5	Euclidean	true	0.854
SonyAIBORobotSurface1	1	0.5	Minkowski	true	0.854
SonyAIBORobotSurface1	5	0.5	Euclidean	true	0.854
SonyAIBORobotSurface1	5	0.5	Minkowski	true	0.854
SonyAIBORobotSurface1	10	0.5	Euclidean	true	0.854
SonyAIBORobotSurface1	10	0.5	Minkowski	true	0.854
SonyAIBORobotSurface2	1	0.5	Euclidean	true	0.866
SonyAIBORobotSurface2	1	0.5	Manhattan	true	0.866
SonyAIBORobotSurface2	1	0.5	Minkowski	true	0.866
SonyAIBORobotSurface2	5	0.5	Euclidean	true	0.866
SonyAIBORobotSurface2	5	0.5	Manhattan	true	0.866
SonyAIBORobotSurface2	5	0.5	Minkowski	true	0.866
SonyAIBORobotSurface2	10	0.5	Euclidean	true	0.866
SonyAIBORobotSurface2	10	0.5	Manhattan	true	0.866
SonyAIBORobotSurface2	10	0.5	Minkowski	true	0.866
Wafer	1	0.25	Euclidean	true	0.994
Wafer	1	0.25	Minkowski	true	0.994
Wafer	5	0.25	Euclidean	true	0.994
Wafer	5	0.25	Minkowski	true	0.994
Wafer	10	0.25	Euclidean	true	0.994
Wafer	10	0.25	Minkowski	true	0.994

dataset	kNeighbors	odSelRatio	odDistFunc	odDescOrder	TP_RATE
Wafer	1	0.5	Chebyshev	true	0.994
Wafer	5	0.5	Chebyshev	true	0.994
Wafer	10	0.5	Chebyshev	true	0.994

Table A.7.: Result parameters for classifier DTW1NN and instance selection method Dist2kNN

dataset	kNeighbors	odSelRatio	odDistFunc	odDescOrder	TP_RATE
Earthquakes	5	0.9	Manhattan	true	0.791
Earthquakes	10	0.9	Manhattan	true	0.791
FordA	1	0.25	Manhattan	false	0.664
ItalyPowerDemand	1	0.9	Euclidean	false	0.976
ItalyPowerDemand	1	0.9	Manhattan	false	0.976
ItalyPowerDemand	1	0.9	Minkowski	false	0.976
ItalyPowerDemand	5	0.9	Chebyshev	false	0.976
ItalyPowerDemand	5	0.9	Euclidean	false	0.976
ItalyPowerDemand	5	0.9	Manhattan	false	0.976
ItalyPowerDemand	5	0.9	Minkowski	false	0.976
ItalyPowerDemand	10	0.9	Manhattan	false	0.976
Lighting2	5	0.05	Manhattan	false	0.885
MoteStrain	5	0.9	Chebyshev	true	0.89
MoteStrain	5	0.9	Manhattan	true	0.89
MoteStrain	10	0.9	Chebyshev	true	0.89
SonyAIBORobotSurface1	1	0.5	Chebyshev	false	0.885
SonyAIBORobotSurface2	1	0.5	Euclidean	false	0.867
SonyAIBORobotSurface2	1	0.5	Minkowski	false	0.867
Wafer	5	0.5	Chebyshev	false	0.995

Table A.8.: Result parameters for classifier DTW1NN and instance selection method LDIS

dataset	kNeighbors	odSelRatio	odDistFunc	odDescOrder	TP_RATE
Earthquakes	1	0.5	Manhattan	false	0.799
Earthquakes	1	0.5	Manhattan	false	0.799
FordA	10	0.25	Minkowski	true	0.655
FordA	10	0.25	Euclidean	true	0.655
FordA	10	0.5	Manhattan	true	0.655
FordA	5	0.5	Manhattan	true	0.655
ItalyPowerDemand	10	0.25	Euclidean	true	0.977
ItalyPowerDemand	10	0.25	Euclidean	true	0.977
ItalyPowerDemand	10	0.25	Minkowski	true	0.977
ItalyPowerDemand	10	0.25	Minkowski	true	0.977
Lighting2	1	0.9	Chebyshev	true	0.869
MoteStrain	1	0.9	Manhattan	false	0.89
SonyAIBORobotSurface1	1	0.5	Chebyshev	true	0.87
SonyAIBORobotSurface1	1	0.5	Chebyshev	true	0.87
SonyAIBORobotSurface2	10	0.5	Chebyshev	true	0.886
Wafer	1	0.05	Euclidean	true	0.995
Wafer	1	0.05	Minkowski	true	0.995
Wafer	5	0.05	Chebyshev	true	0.995

Table A.9.: Result parameters for classifier DTW1NN and instance selection method LKRR

A.5. Literature search

Example of search results for topic time series analysis with the following search terms being used:

- “Time series analysis” OR “Time-series analysis”
- “Time series prediction” OR “Time-series prediction”
- “Time series modeling” OR “Time-series modeling”
- “Time series forecasting” OR “Time-series forecasting”
- “Time series data mining” OR “Time-series data mining”
- “Time series data modeling” OR “Time-series data modeling”

Plattform	Params	#Results	#Relevant
ScienceDirect	Search in: Title, abstract, keywords Article type: Review articles Year >1990 AND (review OR survey OR introduction OR tutorial)	138	7
arXiv.org	Search in: Title, abstract Subject: Computer Science, Mathematics, Statistics Date range: 1990-2018	528	17
DE Gruyter	Search in: Title Subject: Computer Sciences, Business and Economics, Physics 1990-2018	109	1
Scopus	Search field: Document title Date range: 2000-2018 Document type: Article OR Review Subject area: Computer Science	737	11

Plattform	Params	#Results	#Relevant
Web of Science	Search in: Title WEB OF SCIENCE CATEGORIES: (STATISTICS PROBABILITY OR MATHEMATICS OR COMPUTER SCIENCE THEORY METHODS OR COMPUTER SCIENCE SOFTWARE ENGINEERING OR COMPUTER SCIENCE ARTIFICIAL INTELLIGENCE OR COMPUTER SCIENCE INFORMATION SYSTEMS) Timespan: 1900-2018 Document Types: Article, Proceedings paper, Review	618	6
IEEE Xplore	Search in: Document title Content types: Conferences, Journals & Magazines Year Range: 2000-2018	893	18
EmeraldInstight	Search in: Publication title Publication date: 01.2000-12-2018	130	2
SpringerLink	Search field: Title Discipline: Computer Science, Statistics, Mathematics Include Preview-Only content: False	390	7
Google Scholar	with the exact phrase where my words occur: in the title of the article Return articles dated between: 2000-2018	144	6

Plattform	Params	#Results	#Relevant
DOAJ	Search field: Title Subject: Science, Technology, Mathematics, Information Technology	368	4
Recommendations	-	-	6

Table A.10.: Search parameters and results overview for all platforms for the topic ”time series analysis”

Bibliography

[14] 'Data smashing: uncovering lurking order in data'. In: *Journal of The Royal Society Interface* 11.101 (Oct. 2014), pp. 20140826–20140826. ISSN: 1742-5689. DOI: 10.1098/rsif.2014.0826. URL: http://rsif.royalsocietypublishing.org/cgi/doi/10.1098/rsif.2014.0826 (cit. on pp. 1, 2).

[AAB11] Antonio Arauzo-Azofra, Jose Luis Aznarte and Jose M. Benitez. 'Empirical study of feature selection methods based on individual feature evaluation for classification problems'. In: *Expert Systems with Applications* 38.7 (July 2011), pp. 8170–8177. ISSN: 09574174. DOI: 10.1016/j.eswa.2010.12.160. URL: https://hal-mines-paristech.archives-ouvertes.fr/hal-00614451%20https://linkinghub.elsevier.com/retrieve/pii/S095741741001523X (cit. on pp. 1, 13, 42).

[ANH16] Mohiuddin Ahmed, Abdun Naser Mahmood and Jiankun Hu. 'A survey of network anomaly detection techniques'. In: *Journal of Network and Computer Applications* 60 (Jan. 2016), pp. 19–31. ISSN: 10958592. DOI: 10.1016/j.jnca.2015.11.016. arXiv: 55. URL: http://dx.doi.org/10.1016/j.jnca.2015.11.016%20http://linkinghub.elsevier.com/retrieve/pii/S1084804515002891%20https://linkinghub.elsevier.com/retrieve/pii/S1084804515002891 (cit. on pp. 2, 18).

[Bag+17] Anthony Bagnall et al. 'The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances'. In: *Data Mining and Knowledge Discovery* 31.3 (May 2017), pp. 606–660. ISSN: 1384-5810. DOI: 10.1007/s10618-016-0483-9. arXiv: 1602.01711. URL: http://link.springer.com/10.1007/s10618-016-0483-9 (cit. on pp. 11, 38, 39, 43).

[BHK18] Christoph Bergmeir, Rob J. Hyndman and Bonsoo Koo. 'A note on the validity of cross-validation for evaluating autoregressive time series prediction'. In: *Computational Statistics & Data Analysis* 120.April (Apr. 2018), pp. 70–83. ISSN: 01679473. DOI: 10.1016/j.csda.2017.11.003. URL: http://www.sciencedirect.com/science/article/pii/S0167947317302384%20https://linkinghub.elsevier.com/retrieve/pii/S0167947317302384 (cit. on p. 41).

[BL14] Anthony Bagnall and Jason Lines. *An Experimental Evaluation of Nearest Neighbour Time Series Classification*. Tech. rep. June 2014. arXiv: 1406.4757. URL: http://arxiv.org/abs/1406.4757 (cit. on pp. 10, 11, 36, 41).

[BM02] H Brighton and C Mellish. 'Advances in Instance Selection for Instance-Based Learning Algorithms'. In: *Data mining and Knowledge Discovery* 6.2 (2002), pp. 153–172. DOI: 10.1023/A:1014043630878. URL: https://doi.org/10.1023/A:1014043630878 (cit. on p. 11).

[CA15] Joel Luis Carbonera and Mara Abel. 'A Density-Based Approach for Instance Selection'. In: *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, Nov. 2015, pp. 768–774. ISBN: 978-1-5090-0163-7. DOI: 10.1109/ICTAI.2015.114. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7372210 (cit. on p. 41).

[CBK09] Varun Chandola, Arindam Banerjee and Vipin Kumar. 'Anomaly detection'. In: *ACM Computing Surveys* 41.3 (July 2009), pp. 1–58. ISSN: 03600300. DOI: 10.1145/1541880.1541882. URL: http://portal.acm.org/citation.cfm?doid=1541880.1541882 (cit. on pp. 1, 18–21, 28, 41).

[Dag10] EB Dagum. 'Time series modelling and decomposition'. In: *Statistica* 70.4 (2010), pp. 433–457. DOI: 10.6092/issn.1973-2201/3597. URL: http://ideas.repec.org/a/bot/rivsta/v70y2010i4p433-457.html (cit. on pp. 6, 9).

[Dau+18] Hoang Anh Dau et al. 'The UCR Time Series Archive'. In: *CoRR* abs/1810.07758 (2018). arXiv: 1810.07758. URL: http://arxiv.org/abs/1810.07758 (cit. on p. 40).

[EA12] Philippe Esling and Carlos Agon. 'Time-series data mining'. In: *ACM Computing Surveys* 45.1 (Nov. 2012), pp. 1–34. ISSN: 03600300. DOI: 10.1145/2379776.2379788. arXiv: 1710.03346. URL: http://dl.acm.org/citation.cfm?doid=2379776.2379788 (cit. on pp. 8–10).

[FHH16] Eibe Frank, Mark A. Hall and Witten Ian H. *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*. 4th. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016 (cit. on p. 31).

[FPS96] Usama Fayyad, Gregory Piatetsky-Shapiro and Padhraic Smyth. 'From Data Mining to Knowledge Discovery in Databases'. In: *AI Magazine* 17.3 (1996), p. 37. ISSN: 0738-4602. DOI: 10.1609/aimag.v17i3.1230. arXiv: aimag.v17i3.1230 (cit. on p. 8).

[Fu11] Tak-chung Fu. 'A review on time series data mining'. In: *Engineering Applications of Artificial Intelligence* 24.1 (Feb. 2011), pp. 164–181. ISSN: 09521976. DOI: 10.1016/j.engappai.2010.09.007. URL: http://www.sciencedirect.com/science/article/pii/S0952197610001727%20http://linkinghub.elsevier.com/retrieve/pii/S0952197610001727 (cit. on pp. 6, 8–10).

[FV17] Amin Fakhrazari and Hamid Vakilzadian. 'A survey on time series data mining'. In: *2017 IEEE International Conference on Electro Information Technology (EIT)*. Vol. 2. 5. IEEE, May 2017, pp. 476–481. ISBN: 978-1-5090-4767-3. DOI: 10.1109/EIT.2017.8053409. URL: http://ieeexplore.ieee.org/document/8053409/ (cit. on pp. 6, 8–10).

[GE03] Isabelle Guyon and Andre Elisseeff. 'An Introduction to Variable and Feature Selection'. In: *Journal of Machine Learning Research (JMLR)* 3.3 (Nov. 2003), pp. 1157–1182. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=944919.944968 (cit. on pp. 12, 13).

[GLH15] Salvador Garcia, Julian Luengo and Francisco Herrera. 'Instance Selection'. In: *Data Preprocessing in Data Mining*. Cham: Springer International Publishing, 2015, pp. 195–243. ISBN: 978-3-319-10247-4. DOI: 10.1007/978-3-319-10247-4_8. URL: https://doi.org/10.1007/978-3-319-10247-4%7B%5C_%7D8%20http:

//link.springer.com/10.1007/978-3-319-10247-4%7B%5C_%7D8 (cit. on pp. 16, 17).

[Gup+14] Manish Gupta et al. 'Outlier Detection for Temporal Data: A Survey'. In: *IEEE Transactions on Knowledge and Data Engineering* 26.9 (Sept. 2014), pp. 2250–2267. ISSN: 1041-4347. DOI: 10.1109/TKDE.2013.184. arXiv: 0703101v1 [cs]. URL: http://ieeexplore.ieee.org/document/6684530/.

[Guy+06] Isabelle Guyon et al., eds. *Feature Extraction*. Vol. 207. Studies in Fuzziness and Soft Computing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. ISBN: 978-3-540-35487-1. DOI: 10.1007/978-3-540-35488-8. URL: http://link.springer.com/10.1007/978-3-540-35488-8 (cit. on pp. 12–15).

[HA04] Victoria J. Hodge and Jim Austin. 'A Survey of Outlier Detection Methodologies'. In: *Artificial Intelligence Review* 22.2 (Oct. 2004), pp. 85–126. ISSN: 0269-2821. DOI: 10.1007/s10462-004-4304-y. URL: http://link.springer.com/10.1007/s10462-004-4304-y (cit. on pp. 18, 20, 22).

[KK03] Eamonn Keogh and Shruti Kasetty. 'On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration'. In: *Data Mining and Knowledge Discovery* 7.4 (Oct. 2003), pp. 349–371. ISSN: 1573-756X. DOI: 10.1023/A:1024988512476. URL: https://doi.org/10.1023/A:1024988512476 (cit. on p. 39).

[Li+09a] Yang Li et al. 'An Experimental Study on Instance Selection Schemes for Efficient Network Anomaly Detection'. In: *Recent Advances in Intrusion Detection*. Ed. by Engin Kirda, Somesh Jha and Davide Balzarotti. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 346–347. ISBN: 978-3-642-04342-0 (cit. on p. 23).

[Li+09b] Y. Li et al. 'Optimizing Network Anomaly Detection Scheme Using Instance Selection Mechanism'. In: *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*. Nov. 2009, pp. 1–7. DOI: 10.1109/GLOCOM.2009.5425547 (cit. on p. 23).

[Li11] Yuhua Li. 'Selecting training points for one-class support vector machines'. In: *Pattern Recognition Letters* 32.11 (2011), pp. 1517–1522. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2011.04.013. URL: http://www.sciencedirect.com/science/article/pii/S0167865511001255 (cit. on p. 23).

[Olv+10] J. Arturo Olvera-Lopez et al. 'A review of instance selection methods'. In: *Artificial Intelligence Review* 34.2 (Aug. 2010), pp. 133–143. ISSN: 0269-2821. DOI: 10.1007/s10462-010-9165-y. URL: http://link.springer.com/10.1007/s10462-010-9165-y (cit. on pp. 15, 16).

[Pan+16] G. Pang et al. 'Unsupervised Feature Selection for Outlier Detection by Modelling Hierarchical Value-Feature Couplings'. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. Dec. 2016, pp. 410–419. DOI: 10.1109/ICDM.2016.0052.

[PC12] Qinmu Peng and Yiu-Ming Cheung. 'Sample Outlier Detection Based on Local Kernel Regression'. In: *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*. Vol. 1. IEEE, Dec. 2012, pp. 664–668. ISBN: 978-1-4673-6057-9. DOI: 10.1109/WI-IAT.2012.260. URL: http://ieeexplore.ieee.org/document/6511959/ (cit. on p. 24).

[PC14] Qinmu Peng and Yiu-ming Cheung. 'Outlier Detection Based on Local Kernel Regression for Instance Selection'. In: *International Journal of Computational Intelligence Systems* 7.4 (July 2014), pp. 748–757. ISSN: 1875-6891. DOI: 10.1080/18756891.2014.960230. URL: http://www.atlantis-press.com/php/paper-details.php?id=25868516 (cit. on p. 24).

[Pim+14] Marco A.F. Pimentel et al. 'A review of novelty detection'. In: *Signal Processing* 99 (June 2014), pp. 215–249. ISSN: 01651684. DOI: 10.1016/j.sigpro.2013.12.026. URL: http://linkinghub.elsevier.com/retrieve/pii/S016516841300515X%20https://linkinghub.elsevier.com/retrieve/pii/S016516841300515X (cit. on p. 18).

[PM16] Luca Puggini and Sean McLoone. 'Feature Selection for Anomaly Detection Using Optical Emission Spectroscopy'. In: *IFAC-PapersOnLine* 49.5 (2016). 4th IFAC Conference on Intelligent

Control and Automation SciencesICONS 2016, pp. 132–137. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2016.07.102. URL: http://www.sciencedirect.com/science/article/pii/S2405896316302981.

[Rat+09] Chotirat Ann Ratanamahatana et al. 'Mining Time Series Data'. In: *Data Mining and Knowledge Discovery Handbook*. Ed. by Oded Maimon and Lior Rokach. Boston, MA: Springer US, 2009, pp. 1049–1077. ISBN: 978-0-387-09823-4. DOI: 10.1007/978-0-387-09823-4_56. URL: https://doi.org/10.1007/978-0-387-09823-4%7B%5C_%7D56%20http://link.springer.com/10.1007/978-0-387-09823-4%7B%5C_%7D56 (cit. on pp. 7–10).

[RKA06] Juan J. Rodriguez, Ludmila I. Kuncheva and Carlos J. Alonso. 'Rotation Forest: A new classifier ensemble method'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.10 (2006), pp. 1619–1630. ISSN: 0162-8828. URL: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2006.211 (cit. on p. 12).

[SG16] S.Sabeena* and G.Priyadharshini. 'FEATURE SELECTION AND CLASSIFICATION TECHNIQUES IN DATA MINING'. In: *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES AND RESEARCH TECHNOLOGY* 5.4 (Apr. 2016), pp. 160–165. URL: http://doi.org/10.5281/zenodo.48850 (cit. on p. 1).

[SMA12] N. N. R. R. Suri, M. N. Murty and G. Athithan. 'Unsupervised feature selection for outlier detection in categorical data using mutual information'. In: *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*. Dec. 2012, pp. 253–258. DOI: 10.1109/HIS.2012.6421343.

[Sta+19] Darko Stanisavljevic et al. 'Detection of interferences in an additive manufacturing process: an experimental study integrating methods of attribute selection and machine learning'. In: *International Journal of Production Research* (2019). Publication submitted (cit. on p. 31).

[VA17] Vo Thanh Vinh and Duong Tuan Anh. 'Instance reduction for time series classification using MDL principle'. In: *Intelligent Data Analysis* 21.3 (June 2017), pp. 491–514. ISSN: 1088467X. DOI: 10.3233/IDA-150475. URL: http://www.medra.org/servlet/

aliasResolver?alias=iospress%7B%5C&%7Ddoi=10.3233/IDA-150475 (cit. on p. 23).

[VE14] Jorge R Vergara and Pablo A Estévez. 'A review of feature selection methods based on mutual information'. In: *Neural Computing and Applications* 24.1 (Jan. 2014), pp. 175–186. ISSN: 0941-0643. DOI: 10.1007/s00521-013-1368-0. URL: https://doi.org/10.1007/s00521-013-1368-0%20http://link.springer.com/10.1007/s00521-013-1368-0 (cit. on pp. 13, 15).

[Wei04] Gary M. Weiss. 'Mining with rarity'. In: *ACM SIGKDD Explorations Newsletter* 6.1 (June 2004), p. 7. ISSN: 19310145. DOI: 10.1145/1007730.1007734. URL: http://portal.acm.org/citation.cfm?doid=1007730.1007734 (cit. on pp. 2, 45).

[WM00] D Randall Wilson and Tony R Martinez. 'Reduction Techniques for Instance-Based Learning Algorithms'. In: *Machine Learning* 38.3 (2000), pp. 257–286. DOI: 10.1023/A:1007626913721. URL: https://doi.org/10.1023/A:1007626913721 (cit. on p. 16).