Vrije Universiteit Amsterdam

Focus Orange

Detecting Interesting Outliers

Active Learning for Anomaly Detection

June 12, 2016

Author:
Floris den Hengst

Supervisors:
Mark Hoogendoorn
Dirk Jonker

Abstract

We propose an active learning framework for finding anomalies without *a priori* knowledge. The framework consists of an unsupervised component for detection of possible anomalies, a query selection mechanism to minimize the number of required labels and a supervised component for final classification. We argue that LOF is the most suitable unsupervised method as it contains little assumptions on the data. We find that an SVM-based classifier yields the best performance for the supervised. Our approach enables the identification anomalies quickly when all data is available for querying.

Preface

This thesis would not have been possible without excellent guidance, great inspiration and indefinite support from a number of people, of which I'm deeply thankful.

The journey that culminated into this work was initiated by Dirk Jonker with the open-ended question 'How can we detect and errors in customer data?' Although I'm paraphrasing, the lack of a strict definition and any predefined goals in his original statement both posed a challenge and conveyed a clear message of trust. This unusual combination of posing challenges without steering towards paved roads I view as a clear example of 'noise'...

For being an expert in scientific precision, I thank Mark Hoogendoorn and all of his critical questions I can recall. Even though his throughput of papers to review, courses to run and research to conduct kept growing throughout this journey, the unfaded attention and continously insightful remarks proved a firm basis for improvements to both the research and this piece of writing.

Finally, a note of thanks to friends, family and all others close to me: for listening to ramblings and rants; for never-ending status update requests; for providing ever-necessary distractions; for being motivational support machines; for holding up a mirror and keeping my spirits up.

Contents

Chapter 1

Introduction

The analysis of outliers is an important step in any data-intensive task, as deviations from normality can give easy insight into how the data was generated, provide a sanity check and strengthens confidence in further analysis of the data. Any introductory course on statistics or data mining contains a section devoted to the assessment of data quality and strategies for finding and dealing with possibly anomalous instances.

Advancements in technology and a continuously growing adoption of technology by the society at large has resulted in an increasingly growing amount of data on a huge range of topics. This 'new' kind of data can contain highly valuable information, however, it is often highly complex while messy at the same time. In contrast with data being generated in carefully crafted experiments in a way that matches data analysis goals as was the typical scientific method for centuries, current data analysis challenges deal with situations in which such analysis are not applicable – be it because of lack of control over the data generating process, the large volume of data or the exploratory nature of the analysis. Albeit these shifts in the approach to data generation and analysis, anomalies have continued to arise and have continued to be of importance.

At the same time, data-driven decision making is becoming more and more important to organisations. Business functions which have traditionally been tackled from qualitative and ideology-based perspectives, such as customer care and human resource management (HR), have recently started taking an interest in data-driven approaches. These business functions typically deal with the 'new' kind of data that was described above.

Focus Orange is a company that provides data-driven analytics to HR departments on topics such as succession planning, talent management and working conditions preferences. By combining domain knowledge with quantitative methods and an automated approach, they gap the bridge between traditional HR and upcoming technologies.

1.1 Motivation

Recently, Focus Orange has launched 'Crunchr': an online HR analytics platform that brings self-service to the HR analytics domain. It has been observed in practice that a key element in the successful adoption of this platform is the quality of the data that it is based on. Faulty data can harm the analyses in the platform and generally can make results unreliable. It is therefore desired that data that enters the platform contains little errors. This requirement of high quality data, however, is not easily met by most organisations for which Crunchr might be of interest.

It is therefore vital that all data is validated before it enters Crunchr. This is currently achieved by the combined effort of a domain expert and a data scientist: the data scientist's goal is to find possible errors and report these to the domain expert, who then provides the correct values where appropriate. Besides some intuition and rules-of-thumb, the data scientist's main tool in finding these values efficiently and quickly is outlier analysis (also known as extreme value analysis).

This manual approach is time consuming, error prone and goes against the self-service principle of the platform. It requires back-and-forth communication between the domain expert and data scientist, both of whom might struggle with understanding each others point of view and jargon. Furthermore, important insights about a specific organisation might get lost over time: the continuation of knowledge about which unlikely values are erroneous fully depends on the data scientist.

Since organisations use different data management systems, there is no upfront knowledge on which errors might be present. If such upfront knowledge were easily available, the errors would not exist in the first place. Because we cannot know up front which errors are present, each data set has to be audited separately.

Furthermore, the definition of what should be viewed as a (possible) error differs per organisation. For example: the degree of internationalization, organisation size and workforce composition could all contribute to what is seen as an 'abnormal' salary for an organisation. An evidently erroneous situation for the one organisation could be perfectly normal for another. This makes the formulation of a single predefined method (e.g. a set of rules) for finding abnormalities for multiple data set sources infeasible. Another, more data-dependant method is therefore desired.

Although various definitions of the concept 'outlier' exist, they all convey a suspicion about an entry being unlikely when compared to the remainder of the available entries – and thus possibly erroneous [29] [6] [34]. Not all unlikely entries, however, are of special interest. The distinction between 'interesting' and 'not interesting' is by definition domain-specific: it depends on assumptions about the way the data was created (the *generating process* of the data), which features are more important than others and possibly even future usage of the data.

We assume that these interesting and not interesting outliers can be separated somehow: although domain-specific, this distinction cannot be completely arbitrary. If this were the case, a human would not be able to make a proper distinction either. We also assume the data should contain enough information to make this distinction: a human would not require external sources.

Interesting outliers are known as *anomalies* or *strong outliers*, other outliers are usually called *noise* or *weak outliers* [2]. Since this research is aimed at finding errors for a data validation process, the terms *anomaly* for interesting outliers and *noise* for all other outliers seems most appropriate and will be used from here on out. Non-outliers are called *normal* entries.

1.2 Relation to other work

The various usages of outlier detection has lead to a wide variety of definitions of the term 'outlier'. Assumptions on knowledge of generating process, dimensionality, shape and locality of patterns have lead to different *models* for outlier detection: the probabilistic model are backed by statistical theory but require knowledge on the generating processes of the data (underlying distribution, number of subpopulations, etc.), the extreme value is aimed at identifying outliers in the outskirts of the data only, the subspace model requires global correlations in the data, proximity based methods are suitable for clustered data. Selecting the right model is key in the successful selection of an outlier detection method.

From a Machine Learning perspective, methods can be categorized by the availability of labels: Unsupervised outlier detection methods, such as those described in [20], [61], [5] and [9] aid in finding outliers in a data set (discovery of outliers) and are generally optimized for certain uses cases: (a lack of) knowledge on the underlying distribution, a specific data set size or data dimensionality and a focus on numerical or categorical data all play a role in selecting an appropriate method for a use case. Unsupervised methods cannot make a distinction between noise and anomalies, as this distinction is by definition domain-specific.

Supervised methods can make this distinction between anomalies and noise. These methods solve a specific variant of the binary classification problem, in which a large imbalance between the available examples is present. Supervised methods require examples from both classes in a part of the data (the *training set*) from which the difference between anomalies and non-anomalies is learned.

Semi-supervised methods are rather rare in the anomaly detection domain. Most of them are designed to work on a data set in which only examples from one of both classes (anomaly, non-anomaly) are present, whereas in the current setting examples from both classes are present. After a training phase, they are able to detect points that are unlike the previously seen data, which is often referred to as *novelty detection* [30]. Active learning approaches, in which points are selected for labelling by a human, can be found in [70] [47]. The goal here is to select as little points for labelling while still maintaining good performance. These methods can be applied when unaided labelling of points is expensive, however, their approach does not fit our use case.

1.3 Problem Statement

The goal of this research is to develop a method to efficiently separate anomalies from noise and normal entries in a data set. This goal is split up into two sub goals:

1. efficiently detect outliers without *a priori* knowledge
2. separating noise from anomalies among outliers

We aim for a method that can handle the absence of class labels initially, operates on data sets where normal points and noise as well as anomalies may be present and is robust to unknown underlying distributions.

In order to limit the scope of this research, we consider only numerical (continuous and interval) data. The rationale for this is threefold: most features in our use case are numerical; categorical features can be converted into numerical features relatively easy using 'dummy variables', and finally: errors in other kinds of data can be detected relatively easy in other ways [66].

1.4 Key Contributions

This thesis starts out by exploring different definitions of what an outlier is and describes related groups of outlier detection methods, which we call outlier analysis *models*. Both unsupervised and supervised learning approaches are treated, as both are present in the active learning framework we present in Chapter 3. Besides an unsupervised and a supervised method, the proposed framework consists of a selection mechanism which selects the next instance to be labelled by an expert.

We argue that Precision and Recall are the most relevant performance metrics in our use case and we describe how we use them to measure performance on training and test data. These are both relevant as they correspond to the two sub goals presented in the previous section. We find that a combination of Local Outlier Factor and C-SVC yields the best performance and that our query selection mechanism improves random querying. Applying a trained model on previously unseen data does not yield better results than a simple unsupervised method. Investigating whether an ensemble of unsupervised models yields better scores is left for future work.

Chapter 2

Related Work

Outliers can be analysed to remove erroneous points in a data set. As such, outlier detection is a basic step in many data validation processes and a prerequisite to doing statistical analysis on data of which the presence of outliers is unknown. Outliers, ways to find outliers and methods to analyse outliers have been studied extensively as a result.

Historically, this study was focused on estimating parameters of underlying distributions of observations. With the rise of information technology, advances in statistics and growth of both data and computational power a wide variety of outlier analysis methodologies have been developed. Outlier detection is used in various domains, ranging from finance and health care to astrophysics, and has now surpassed its original use as a preparation step for further statistical analysis. Applications in intrusion detection for web servers [50], finding brain tumours from MRI scans [51] and fraud detection [48] exemplify this.

Alongside this growth in outlier detection applications, a multitude of different definitions of the concept outlier has been developed. For example, in [27] we find:

> An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs

Hawkins links outliers to the notion of a generating process [29]:

> An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.

With differing definitions of what an outlier is, naturally come fundamentally different approaches to outlier detection. The terms outlier and outlier detection should therefore be viewed as collective terms in which actually a multitude of different concepts is contained.

In this chapter, we will give an overview of the available methods for outlier analysis. In the first section we compare some outlier detection methods suitable which can be used for exploratory purposes: when no labelled instances are present, an assumption on what constitutes an outlier should be made. This leads to various different *models* of outlier analysis in the *unsupervised scenario*: probabilistic, extreme value, subspace- and proximity-based models are described with some exemplary methods.

When available, including information from known normal points and anomalies into the method can be beneficial. We treat three possible variations in the *supervised* scenario: fully supervised, semi-supervised and active learning and show how they relate to our use case as described in Section 1.3.

Most techniques are presented separately for the sake of a comprehensible overview. In reality, however, these techniques are combined and arranged in multiple ways to improve performance and robustness. We will see some examples of this as well throughout the discussion of the methods in this chapter.

2.1 Unsupervised Scenario

When there are no labelled outliers available, the outlier detection task is of an exploratory nature. This *unsupervised* scenario requires methods that can find outliers without human supervision. Because of the absence of exemplary outliers, an assumption has to be made on what separates outliers from normal points. Depending on the data, an appropriate definition of the term 'outlier' should be used. These different definitions have resulted in a wide variety of fundamentally different ways to find outliers: these can be regarded as different *models* of outlier detection.

We will start by having a look at the probabilistic model, which is derived from statistical theory and treats points that are unlikely given an underlying distribution as outliers. A more basic model is treated next in which outliers lie at the edge of the data: the extreme value model. The subspace model is based on dependence between features and denotes points which cannot be represented in a lower-dimension space properly as outliers. The last model in this section is the proximity-based model. It treats points that are far away from other points as outliers.

2.1.1 Probabilistic Model

The field of outlier detection originates in statistics: it was originally associated with identifying observations which are unlikely to stem from an assumed underlying distribution. These deviations from the expected distribution may be attributable to measurement error and it might therefore be wise to exclude them from further analysis. These methods are also known as probabilistic, statistical or distribution-based [2] [70].

One of the earliest known mentions of outlier detection, *Chauvenets criterion*, can be applied to find outliers in a univariate set of observations drawn from normally distributed populations [11], for instance, these 10 observations: $\{-2, 6, 9, 9, 10, 11, 11, 13, 16, 18\}$. First, the observed mean μ and standard deviation σ are calculated. In our example, $\mu = 10.1$ and $\sigma \approx 5.5$. The value

-2 differs slightly more than 2σ from the observed μ. The probability of this happening is roughly 0.03 given the theoretical normal distribution. The criterion states that we should reject every data point for which the probability of occurring given the observed mean and sigma is less than $\frac{1}{2n}$, so we should suspect all probabilities smaller than 0.05 in our example. This means we would reject the suspected point according to this criterion. A general approach based on QQ-plots that supports non-normal underlying distributions can be found in [65]. First, the distribution of the data is approximated by regression of the observed values on their estimated QQ-plot positions. Next, a lower and upper threshold are determined between which a certain percentage of all points must lie (e.g. 5%). All points below and above the lower and upper threshold respectively are reported as outliers.

With the use of *mixture models* this simple case can be extended to a more advanced one in which a single feature is observed in instances from multiple subpopulations. The subpopulations are assumed to stem from distributions with differing parameters. When we know the number of subpopulations and the underlying distribution of these subpopulations, we can estimate their parameters (e.g. μ, σ or other parameters) by using the *Expectation Maximization* (EM) algorithm [43]. EM starts with randomly parameters for all subpopulations and proceeds in two steps for various rounds: in the E-step each point is assigned to the subpopulation it is expected to be part of based on membership probability. In the M-step, the parameters of the formed subpopulations are adjusted to describe their members best. The algorithm ends on convergence. The different subpopulations are generally known as the *components* of the mixture. Outliers are points that have low membership probabilities for the component they are assigned to (low likelihood) or points for which it is hard to decide to which component they belong (uncertainty) after convergence [57] [41]. Alternating both criteria (*interleave*) yields the best results [47].

This mixture model approach using EM also works for multivariate normally distributed data. Alternatively, plotting the Mahalanobis distances for each point to the mean in a χ^2 QQ-plot visually indicates which points are outliers [25]. We will not go into too much detail, but provide a brief description. The population mean can be estimated from the observed data, and is composed of the mean for each variable, i.e. it is the centre of the data. The Mahalanobis distances along each variable are calculated and summed. The Mahalanobis distance corrects for covariance, so that extremity in two correlated dimensions contributes only once. The distribution of the squares of these summed Mahalanobis distances to the data centre is expected to follow the χ^2 distribution when the data follows a multivariate normal distribution. When plotting the resulting squared summed Mahalanobis distances against the theoretical χ^2 distribution in a QQ-plot, outliers can be found by identifying points that to disturb a straight line. A related, yet automated approach is described in [22]. All of these methods require the underlying distribution to be known.

In conclusion: although they have a solid theoretical foundation, most of the methods based on the probabilistic model require the underlying distribution to be known. Finding the distribution (including correct parameters) is complex: efficient approaches exist [38], but bring their own complexity via the introduction of new parameters. All of these do not apply to many real-life scenarios. Because of these limitations, pure probabilistic-based methods are not often used for outlier detection on high-dimensional data in an automated setting.

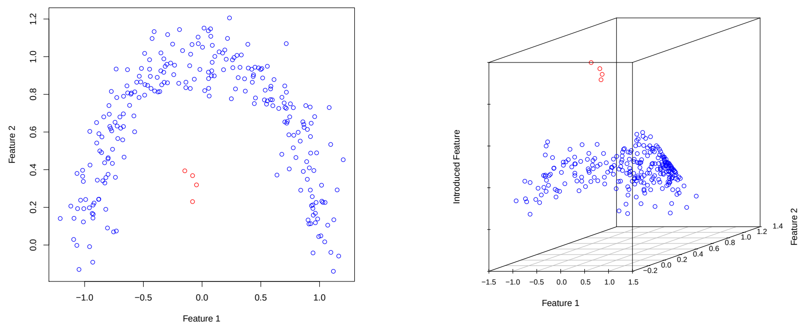
2.1.2 Extreme Value Model

A theoretically more basic model for identifying outliers is by inspection of values in the outer range of a set of data points, commonly known as *extreme value analysis* [49]. Extreme value analysis in the univariate case simply returns the top n points furthest from the centre of the data, i.e. the highest and lowest values in the set. When all samples stem from the same population, this model is very similar to the non-mixture probabilistic model in that the outliers always lie in the *outskirts* of a data set. The difference between both models becomes clear when considering a probabilistic mixture model. A distinguishing example of can be found in [2]:

> For example, in the data set $\{1, 2, 2, 50, 98, 98, 99\}$ of 1-dimensional values, the values 1 and 99, could very mildly, be considered extreme values. [...] most probabilistic and density-based models would classify the value 50 as the strongest outlier in the data, on the basis of Hawkins' definition of generative probabilities.

A mixture model consisting of two components is expected to fit the three values on the left and right respectively to finally find that value 50 is an outlier as it does not fit either of the components very well. Extreme value methods would not consider 50 to be an outlier, as it resides at the centre of the set. Extreme value methods are expected to return values 1 and 99, as these are most distant to the centre of the data. Extreme value analysis can be extended to a multivariate setting in various ways. *Depth-based* methods try to fit multiple convex hulls (or *contours*) to the data [55] [35]. Outliers can then be found by 'peeling' off the outer points until a pre-specified amount of data points at the edge of the set has been found. When this pre-specified amount is not known, all points can be ranked based on the *depth* (i.e. when the point was peeled off). This requires the method to compute depth of all data points, which is computationally expensive. These methods are therefore generally not used for datasets with more than 4 features [2]. Although being amongst the first in multidimensional extreme value analysis, there are very little applications because of this restriction.

An extreme-value method that can also detect outliers in the centre of a data set can be found in [13]. It does so by employing a mapping of the original features in the so-called *input space* to a higher-dimensional *feature space*. The mapping should be designed to scatter the data such that formerly central data points move to the outskirts of the data. An example of such a mapping can be found in Figure 2.1, where in the original data the suspected outliers lie at the centre (making them impossible to detect using extreme value methods),



(a) Example dataset with a half-moon shape. Suspected outliers lie in the centre of the data set.

(b) Mapping of Figure 2.1a to higher-dimensional space using a *Radial Basis* function. The suspected outliers now lie in an outskirt of the dataset.

Figure 2.1: Applying a *kernel* can move outliers to outskirts of the dataset.

whereas they lie in the outskirts after the mapping has been applied. The group of functions that can be used is generally known as *kernel functions* or simply kernels. By looking for outliers in feature space rather than in input space, it is possible to find otherwise hidden outliers. When an outlier detection method can be expressed in terms of dot-products on vectors, the kernel does not have to be applied explicitly as it can be included in all dot-product calculations so that the mapping is performed implicitly. This generally does not add any computational complexity [8].

A method suitable for extreme value analysis in high dimensional data can be found in [37]. The idea is that when we draw arrows from a point to its neighbouring points, the outgoing arrows for points with an extreme value are expected to point at a more similar direction than points nearer to the centre of the data set.

This is implemented by calculating the angle of pairs of *difference vectors* from each data point to pairs of neighbouring points. When we consider every point in the set as a vector, we can make a triple of vectors $(\vec{A}, \vec{B}, \vec{C})$ for all combinations of points in the dataset (where $\vec{A} \neq \vec{B} \neq \vec{C}$). The difference vectors $\vec{AB} = \vec{B} - \vec{A}$ and $\vec{AC} = \vec{C} - \vec{A}$ now describe segments from $\vec{A}$ to $\vec{B}$ and from $\vec{A}$ to $\vec{C}$ respectively. See Figure 2.2 for a visual example, where difference vectors are denoted as dotted arrows. Angles formed by pairs of arrows originating from the leftmost point are all acute (note that reflex angles are not taken into account – their acute counterpart is used instead), whereas the angles formed by pairs of arrows originating from a central point are both acute and obtuse. In order to let points close by be of more influence, the angle between these difference vectors is weighted by their length (i.e. the distance

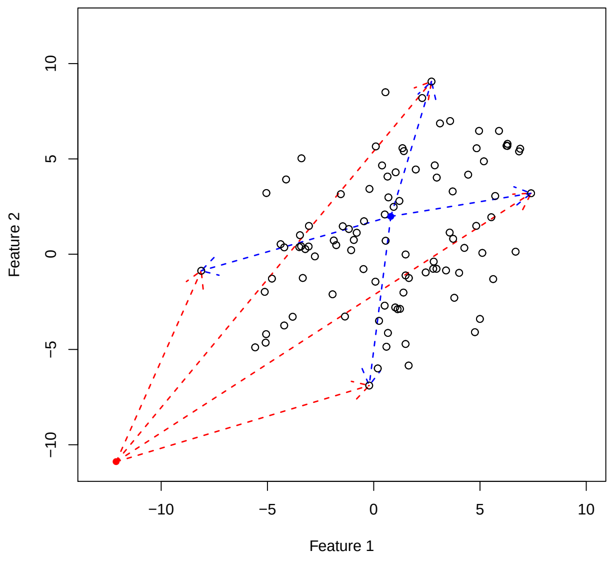


Figure 2.2: Visualisation of the difference vectors used in *Angle-Based Outlier Detection*. The variance of angles between pairs of difference vectors originating for suspected outliers (e.g. the instance in the lower left corner denoted by the red colour) is smaller than the variance of angles between pairs of difference vectors originating from a point in the centre of the data (denoted in blue).

between the points). This weighted angle α is defined as:

$$\alpha = \frac{\langle \vec{AB}, \vec{AC} \rangle}{\left\| \vec{AB} \right\|^2 \times \left\| \vec{AC} \right\|^2} \tag{2.1}$$

where $\langle \vec{AB}, \vec{AC} \rangle$ denotes the dot product of $\vec{AB}$ and $\vec{AC}$ and $\left\| \vec{X} \right\|$ the *norm* (or length) of $\vec{X}$. Note that the weighing is done by squaring the norm of the difference vectors in the denominator. The variance in the weighted angles for $\vec{A}$ to all other points $\vec{B}$ and $\vec{C}$ in the set is expected to be lower for points that lie at the boundaries of the data compared to points that lie closer to the centre. As a performance optimization, only k nearest neighbours of $\vec{A}$ can be sampled for $\vec{B}$ and $\vec{C}$.

The basic extreme value model is somewhat simple in the sense that it only identifies points at the boundary of the data as outlier. *Kernels* can be used to adapt pure extreme value methods to also find outliers in the centre of a data set. Extreme value methods can be computationally hard for high-dimensional data, as they require computation of the ‘outermost’ points of the data. In some

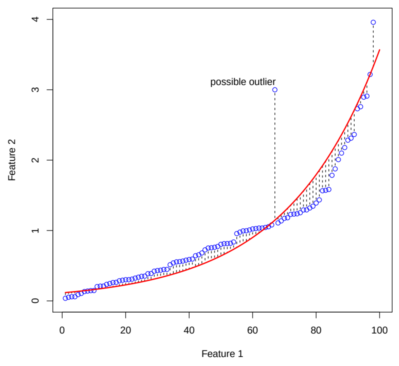


Figure 2.3: A subspace method in which a curve is fitted through a set of data points. Applying extreme value analysis to the residues (dashed lines) of this fitted model can detect outliers at a global scale.

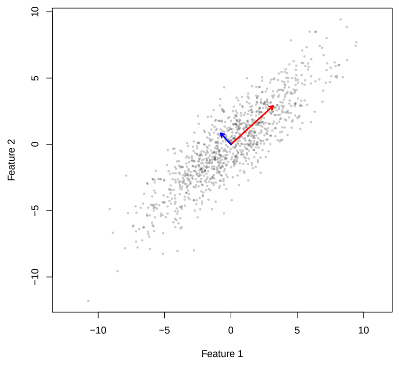


Figure 2.4: PCA on a multivariate normally distributed data set. The arrow pointing to the upper right corner (red) denotes the first principal component. The other arrow (blue) denotes the second principal component.

cases, sampling solves this issue at the cost of a loss of precision. Extreme value analysis on high-dimensional data does not only suffer from performance issues, however: when the amount of dimensions in a data set increases, the number of points at the edge of the data is expected to grow as well. Extreme value methods are therefore not often used on high-dimensional data. However, they often form a crucial final step for methods from other outlier detection models. Consider some method which assigns a score to each data point to express the degree of 'being an outlier'. Extreme value analysis on the righter tail of these scores can aid in selecting data points which should be flagged as outlier.

2.1.3 Subspace Model

An intuitively appealing method to handle the curse of dimensionality from which most probabilistic and extreme value approaches suffer is to reduce the number of dimensions. An appropriate way to do this whilst removing as little information from the data as possible, is by finding a way to express features in terms of a subset of the original set of features.

Consider the two-dimensional dataset drawn from an exponential distribution presented in Figure 2.3. The fitted line can be found by using an appropriate regression technique (e.g. Generalized Linear Model for arbitrarily distributed response variables, logistic regression for categorical data, etc.) and produces a per-point error known as the *residue*. It is visually clear that the point marked as possible outlier has a relatively large residue. Applying an extreme value- or probabilistic-based outlier detection method can identify outlying points. This regression-based approach can be used to detect outliers in multivariate settings as well, but it can find outliers in only one feature at a time, since regression takes only one response variable.

The basic idea described above can be extended to a truly multivariate case by using Principal Component Analysis (PCA). Whereas regression techniques map the data from dimensionality d to dimensionality $d-1$ (i.e. remove 1 feature), PCA can reduce the dimensionality to any $d-k$ whilst accounting for a large deal of variance in the data. Although a complete description is outside of the scope of this research, we will give a short description of PCA for outlier detection: in PCA, a covariance matrix $\sum$ describing the covariances between all features is computed. This covariance matrix is symmetric by the symmetry of covariance and positive semi-definite (i.e. non-negative). Because of these properties, it is possible to decompose this covariance matrix into an orthonormal matrix of eigenvectors P and a diagonal matrix of eigenvalues D:

$$\sum = P \cdot D \cdot P^T \tag{2.2}$$

As the eigenvectors are orthonormal (i.e. mutually orthogonal), the data set can be rotated so that these eigenvectors form a new coordinate system. The eigenvalues corresponding to the eigenvectors are related to the variance along their eigenvector. By taking the k eigenvectors with the highest eigenvalues and rotating the points so that they are aligned along these eigenvectors, the dimensionality can be reduced by $d-k$ whereby the eigenvectors are the axes of the new coordinate system. For example, in Figure 2.4 most variance would be captured along the red arrow (pointing to the top right). It would be possible to rotate the entire dataset so that this red arrow forms the new x-axis and drop the new y-axis (blue arrow) altogether. Extreme value- or probabilistic-based methods can then be applied on the distances between the points in the reduced-dimensional space and the input space to find points that do not follow global patterns. This distance from projected space to input space can be regarded as a measure of 'error' of the mapping for a specific point, which is comparable to taking the residues in regression as described above.

Methods from the subspace model are only suitable when there are globally consistent correlations in the data. When there are no correlations, the mapping will have a large error for all points, making it impossible to detect outliers. When correlations are present, but not globally consistent as in Figure 2.5, the methods will (possibly falsely) report outliers in sections of the data that do not follow globally dominant correlations. When the data contains non-linear correlations, some preprocessing has to be applied or a more intricate model has to be used (such as the GLM for regression).

2.1.4 Proximity-based Model

When the data contains patterns that are not globally consistent, the *proximity based* model is more appropriate. This model relies on the notion of *distance* to define outliers: if a point is located far away from other data points, it might be an outlier. The intuition is that neighbouring points can be expected to be alike and that it is unlikely that points behave differently from neighbouring points. The proximity-based model is appropriate when points are clustered, for example as in Figure 2.5.

Proximity-based methods were first proposed in [36], since which a wide variety of variants has been proposed. Proximity-based methods are popular because of high interpretability and ease of implementation [2]. There are different ways to define 'proximity' of points and multiple ways to flag points based on proximity in comparison to other points. Some exemplary methods are discussed next.

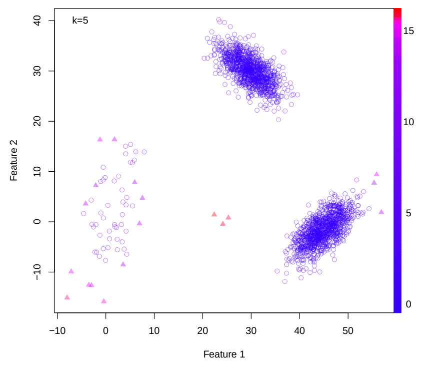
A first example of a proximity-based method is called *Distance-based Outlier* and was proposed in [36]:

> An object O in a dataset T is a $DB(p, D)$-outlier if at least fraction p of the objects in T lies greater than distance D from O.

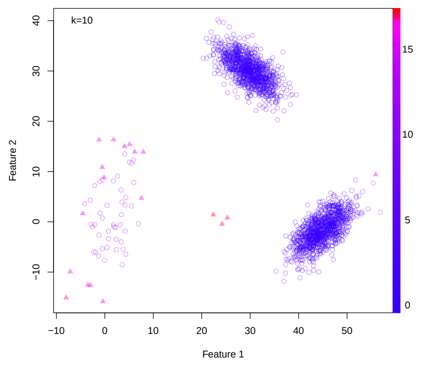
An easier, but similar formulation is often preferred over this original formulation and is generally known as the *k-nearest neighbour approach*. In this formulation, an outlier score is constructed based on the distance to the kth-nearest neighbour. The distance can be used directly as an outlier score, or some transformation (e.g. nonlinear scaling) can be applied. Alternatively, the average distance to the k-nearest neighbours can be used. The distance D in the original formulation can be seen as a threshold value which can also be used in the new formulation, i.e. by discarding all points that have an outlier score below this D. The new formulation ranks all points instead of providing a binary label and thus also allows for reporting a fraction (i.e. top n%) of all points.

K-nearest neighbour methods regard all outliers at a *global* scale, that is: the outlier score of neighbouring points is not taken into account. This makes them not robust to differences in *local density*: when the data is distributed unevenly, the points in the sparse regions are reported more often than desirable. Consider Figure 2.5: here we find three clusters and some suspected outliers in the centre. The cluster in the bottom left, however, contains points that also have few other points close-by, which makes them outliers when viewed at a global scale. However, they will generally be viewed as non-outliers by humans because of the general sparsity in this section of the data. Methods that take local density into account will be discussed next.

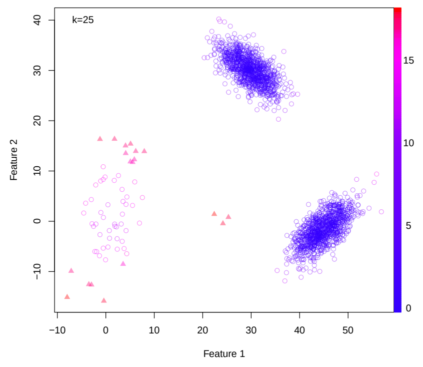
Density-based methods share the notion of proximity with distance-based methods, but include the 'outlieriness' of points in a region around the point before assigning a final outlier score. The intuition is that when the outlieriness of points in a region is equally low, this is probably just a sparse region – which would indicate that these points do not behave unexpectedly and should not be reported as outliers. Density-based methods thus partition the space, in contrast to the distance-based methods that partition individual points. The main goals are dealing with the differences in local density and lowering computational complexity. The two most popular methods are discussed.



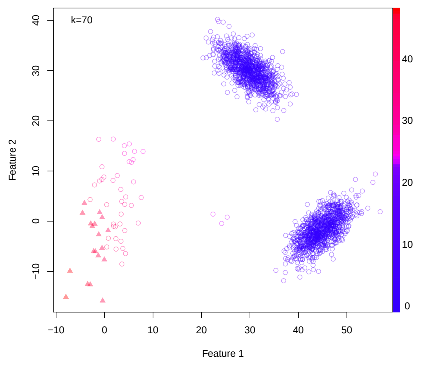
(a) $k = 5$: the points in the centre are correctly detected, along with some erroneously reported points in the lower left cluster.



(b) $k = 10$: the points in the centre are correctly detected. The remainder of detected outliers is primarily located at the outskirts of the lower left cluster.

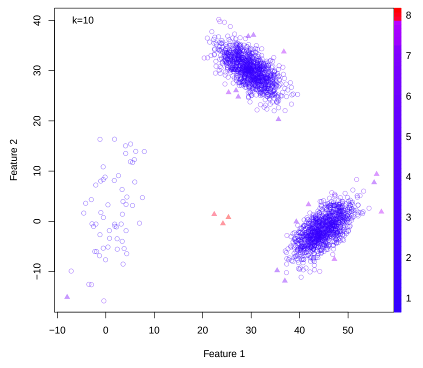


(c) $k = 25$: the points in the centre are correctly detected, however they do not have the highest outlier scores. The remainder of detected outliers are all located at the outskirts of the lower left cluster.

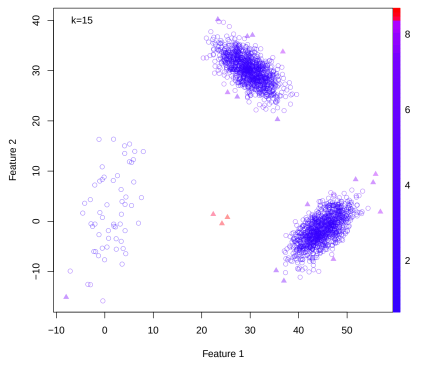


(d) $k = 70$: the points in the centre are no longer in the top 1% of outlier scores. The detected points are all in the lower left corner.

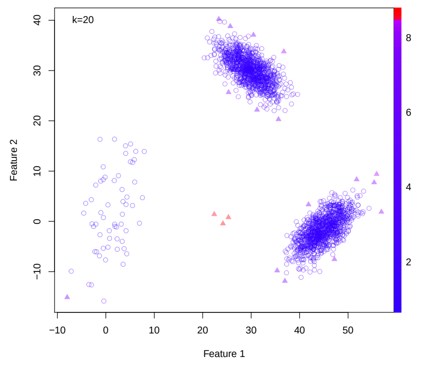
Figure 2.5: Results of the distance-based k-nearest neighbour method on a clustered data set containing some outliers. The top 1% of outlier scores is denoted by triangles. Outlier detection is performed at a global scale (compare: Figure 2.6).



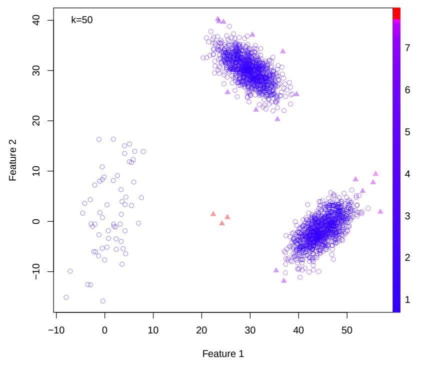
(a) $k = 10$



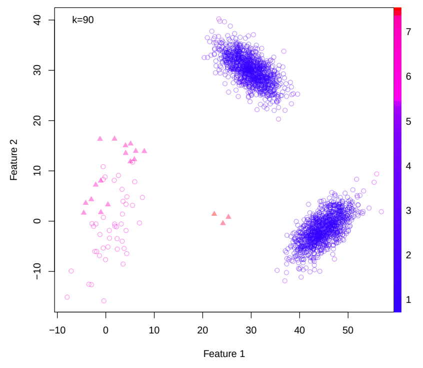
(b) $k = 15$



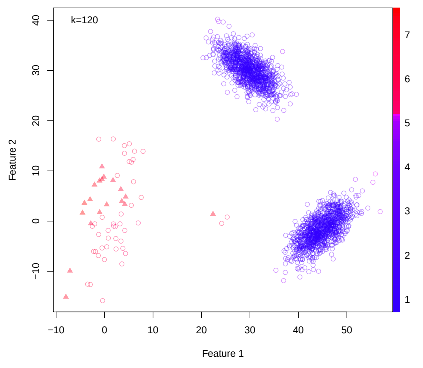
(c) $k = 20$



(d) $k = 50$



(e) $k = 100$



(f) $k = 120$

Figure 2.6: Outlier scores for LOF for differing values of parameter k. The top 1% of scores is denoted by a triangle. LOF proves robust against differences in local density. When k is bigger than the size of a non-outlier cluster, performance degrades (subfigures e and f).

Local-Outlier Factor (LOF) was introduced as a first density-based method in [9] and has met some popularity. It uses the notion of *reachability distance* of a point i from another point j in order to define a per-point outlier score. The reachability distance is defined as:

$$R^k(i,j) = \max(\text{dist}(i,j), D_j^k) \tag{2.3}$$

where D_j^k is the distance from point j to its kth nearest neighbour. This can be interpreted as follows: the reachability distance of i from j is the distance between these points *unless* i is closer to j than j's k-nearest neighbour. All points close to j thus get the same reachability distance. Note that by the above definition, $R^k(i,j)$ is not necessarily the same as $R^k(j,i)$. The points close to j (i.e. closer to j than its kth nearest neighbour) form the *locality* L_i^k of i. The presence of j in i's locality does not necessitate the presence of i in j's locality. The *local reachability distance* is defined as:

$$\text{lrd}(i) = 1/\left(\frac{\sum_{\ell \in L_i^k} R^k(i,\ell)}{|L_i^k|}\right) \tag{2.4}$$

so, as one divided by the average reachability distance of i from all ℓ within i's locality. This *lrd* is used to define the local outlier factor as follows:

$$\text{LOF}_k(i) = \frac{\sum_{\ell \in L_i^k} \frac{\text{lrd}(\ell)}{\text{lrd}(i)}}{|L_i^k|} \tag{2.5}$$

The reachability distances of points in i's locality thus contribute to i's outlier score: when i's local reachability distance is low relative to the reachability distance of points in its locality, then that i is relatively far away from a dense cluster. This happens when the reachability distance of i from some ℓs is not the same as the reachability distance of these ℓs from i and/or when some ℓs are in i's locality without i being in these ℓ's locality.

Rather than selecting a single value for the only parameter k, the authors recommend to compute the LOF for a range of values for k. The final outlier score can then be found by selecting the maximum LOF for the ks in this range. The lower limit for the range can be regarded as the minimum amount of points necessary to constitute a cluster. The upper limit for the range can be viewed as the maximum number of points that may be in a cluster whilst still being regarded as outliers. An example of the result for different values of k on the same dataset can be found in Figure 2.6. Outlier scores are denoted by colour. For lower values of k, the points in the centre are consistently reported as most outlying, along with some points at the outskirts of the clusters. Performance degrades as inappropriate values for k are used ($k \geq 100$).

A similar method, called *local correlation integral* (LOCI) is presented in [45]. The method is presented as superior to LOF due to the elimination of choice for the range of parameter k, but its performance is not guaranteed to be better [32]. LOCI does not require any selection of parameters and the notion of neighbourhood is used more directly in LOCI when compared to LOF.

LOCI uses two notions of neighbourhood: the *sampling neighbourhood* $\mathcal{N}_{sampling}(i, r)$ and the *counting neighbourhood* $\mathcal{N}_{counting}(i, \alpha, r)$. $\mathcal{N}_{sampling}(i, r)$ contains all points within radius r around point i and $\mathcal{N}_{counting}(i, \alpha, r)$ contains all points within radius αr around i. The authors advise to take $0 < \alpha < 1$ so that the radius used for $\mathcal{N}_{sampling}$ is always larger than the radius used for $\mathcal{N}_{counting}$. Let $n_{counting}(i, \alpha, r)$ denote the number of points in $\mathcal{N}_{counting}(i, \alpha, r)$ (including i itself) and $n_{sampling}(i, r)$ denote the number of points in $\mathcal{N}_{sampling}(i, r)$ (again including i itself).

The *average counting neighbourhood size* for all points within $\mathcal{N}_{sampling}(i, r)$ is then defined as:

$$\hat{n}(i, \alpha, r) = \frac{\sum_{j \in \mathcal{N}_{sampling}(i,r)} n_{counting}(i, \alpha, r)}{n_{sampling}(i, r)} \tag{2.6}$$

This average counting neighbourhood size is used to define the multi-granularity deviation factor (MDEF):

$$\text{MDEF}(i, \alpha, r) = 1 - \frac{n_{counting}(i, \alpha, r)}{\hat{n}(i, \alpha, r)} \tag{2.7}$$

The MDEF can be seen as a measure of local density compared to the local densities of points in an extended neighbourhood. The normalized standard deviation for this MDEF is defined as follows:

$$\sigma_{\text{MDEF}(i,\alpha,r)} = \frac{\sigma_{\hat{n}(i,\alpha,r)}}{\hat{n}(i, \alpha, r)} \tag{2.8}$$

where $\sigma_{\hat{n}(i,\alpha,r)}$ denotes the standard deviation of $n_{counting}(j, \alpha, r)$ for all $j \in \mathcal{N}_{sampling}(i, r)$.

These MDEF and σ_{MDEF} are computed over a range of relevant radii r, so that the outlier factor on multiple scales can be found. The suggested range of relevant radii $[r_{\min}, r_{\max}]$ is found by taking an $r_{\min}$ so that $\min(\hat{n}(i, \alpha, r_{\min})) \approx 20$ and $r_{\max}$ corresponds to the maximum $\mathcal{N}_{counting}$ possible in the data set, i.e. by using the distance of the two points with the highest pairwise distance. The rationale for $r_{\min}$ is that this number of observations is required to get a statistically large enough sample when comparing neighbourhood counts, whereas the choice for $r_{\max}$ is dictated by the data: a larger $r_{\max}$ would not lead to new MDEF values. For parameter α, the authors propose two values: $\alpha = 1/2$ for exact computations and $\alpha = 1/16$ for approximations.

Outliers are defined as points for which $\text{MDEF}(i, \alpha, r) > k\sigma_{\text{MDEF}(i,\alpha,r)}$ for any r and with $k = 3$, so for all data points for which the MDEF is more than three times the standard deviation within any r. As noted by the authors, the choice of $k = 3$ theoretically bounds the number of reported points to 10% of the data set, regardless of the distributions in the sampling neighbourhoods, by Chebyshev's inequality.

Proximity-based methods are suitable when applied to clustered data. Distance-based methods have a high granularity, but come with a worst-case complexity of $\mathcal{O}(n^2)$ in a naive implementation. This can be improved in practice through pruning (i.e. stop calculation of distances for points once they are known be a normal point) and through the usage of spatial indexing [36]. Density-based methods partition space rather than individual points: they have a more coarse granularity and are more suitable for data with varying densities. Both types of methods suffer from the curse of dimensionality in the quality of the reported outliers: when the number of dimensions is large, points start to lie at a similar distance making it hard to detect true outliers [3].

2.2 Supervised Scenario

In contrast to the scenario described in the previous section, labels are available to some extent in a *supervised* scenario. The use of labelled instances can improve performance drastically when employed correctly. In this section, we will look at some methods that can be used when all instances are labelled (fully supervised), when some of the instances are labelled (semi-supervised) or can be obtained by interaction with an expert (active learning). Whereas the goal in unsupervised outlier detection is to find outliers based on some implicit definition (model) of the term 'outlier', the goal of its supervised counterpart is to elicit a descriptive pattern that captures the difference between normal points and anomalies – a 'definition' – from examples.

The field of Machine Learning (ML) is associated with finding patterns in data [7]. Mitchell gives a formal definition in [42]:

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Supervised anomaly detection is a 'class of tasks'. Some human input is required to separate between noise and anomalies due to the domain-specific nature of this separation (see Section 1.1). When combined with previous output, this input can be seen as 'experience'. Obviously, we aim for the performance of the method to increase as the number of inputs (i.e. the experience) increases.

Out of the typical ML tasks, supervised anomaly detection resembles the classification task most. A reduction from anomaly detection to classification is provided in [1]. The main differences originate from the importance of class imbalances. Whereas in anomaly detection the domination of normal points over outliers is a central part of the problem definition, it is viewed as a possible complicating factor in general classification. Besides this, however, there are no fundamental differences between classification and supervised anomaly detection.

This section provides an overview of anomaly detection with supervision. First we treat some methods that require class labels for *all* instances to be present. These *fully supervised* methods often form the basis for other methods which can handle label absences as well. The supervised scenario does not fit our use case well, so the overview is not comprehensive. We will continue with some exemplary methods that *can* handle the absence of labels. These *semi-supervised* methods are closely related to methods that actively search for instance of which the labels would improve performance most. These *active learning* methods are suitable when an expert is available, as in our use case.

2.2.1 Fully Supervised

In the fully supervised scenario labels for *all instances* are available initially. When employed correctly, available class labels can improve performance drastically, because of the domain-specific definition of 'anomaly' [2]. Virtually all supervised classification methods can be transformed into anomaly detection methods via these adaptations:

Misclassification costs since most classifiers try to minimize some error function, class imbalance can lead to classifiers that label all points as the majority class. A *cost* of misclassification can be introduced to this error function [59] [62]. By penalizing misclassifications of the minority class more than misclassifications of the majority class, a bias towards labelling as the minority class can be introduced. The number of true positives is expected to increase at the cost of a rise in number of false positives. Some examples of including costs to move decision boundaries for specific algorithm as well as a classifier-independent are discussed next.

Naive Bayes classifiers can provide class probabilities for all classes given an instance. In the general case, the class with the highest probability for this instance is selected. This decision rule can be altered to prefer one class over the other by multiplying the probabilities by some factor c for the anomaly class and some factor $1 - c$ for the normal class [40]. When we prefer more true positives at the expense of some false positives, we can choose $c > 0.5$.

Decision trees recursively split the data set in an 'optimal' way according to some criterion by providing a series of attribute-value tests (e.g. 'feature 1 $>$ 2.0'). The attribute-value tests form the internal nodes of the tree. When a stop condition is reached, a leaf node is created in which class probabilities for all data points are determined by their presence in this part of the split. An instance is classified in two steps: first, the tree is traversed from the root to a leaf node by following the path denoted by the splits in internal nodes. When a leaf node is reached, the class with highest probability in this leaf node forms the prediction for the new instance. The criterion (*Gini impurity*, information gain) used can often be weighted to influence the 'optimal' split. For instance, in [12], the Gini impurity is weighted to influence the final decision in leaf nodes (i.e. shifting class probabilities as in Naive Bayes).

Support Vector Machines (SVMs) can find an optimal linear hyperplane separating two classes (i.e. the decision boundary). They can be formulated in terms of dot products and are therefore suitable for applying *kernels*, to enable non-linear decision boundaries in input space (see above).

For additional details on kernels, see section 2.1.2. They have become popular because of the uncommon ability to find an optimal result. A full description is outside of the scope of this research, but a relatively large description is required to outline the application of misclassification costs in SVMs.

The original formulation of the SVM can be found in [14]. Consider a training set of points $x_1, \ldots, x_i, \ldots, x_n$, and their classes $y_1, \ldots, y_i, \ldots, y_n$ where $y_i = -1$ or $y_i = 1$ for both classes. All separating hyperplanes satisfy

$$\langle w, x \rangle - b = 0 \tag{2.9}$$

where w denotes the normal vector to the hyperplane (i.e. a vector perpendicular to the hyperplane), b denotes a constant and $\langle A, B \rangle$ denotes the dot product of A and B. Assuming linear separability, the distance between two hyperplanes should be maximized to find the optimal separating hyperplane:

$$\langle w, x \rangle - b = 1 \tag{2.10}$$

$$\langle w, x \rangle - b = -1 \tag{2.11}$$

The distance between these hyperplanes is $\frac{2}{\|w\|}$, so the goal becomes to minimize $\|w\|$. Note that $\|w\| = \sqrt{\langle w, w \rangle}$ when $w \in \mathbb{R}^n$, so that the object of minimization can be simplified to $\langle w, w \rangle$. In order to prevent training instances to fall into the margin, the following constraints are added:

$$\begin{aligned} \langle w, x_i \rangle - b &\geq 1 && \text{for } y_i = 1 \\ \langle w, x_i \rangle - b &\leq -1 && \text{for } y_i = -1 \end{aligned} \tag{2.12}$$

which can be rewritten as

$$y_i(\langle w, x_i \rangle - b) \geq 1 \tag{2.13}$$

The optimization goal then becomes:

$$\begin{aligned} &\text{minimize} && \langle w, w \rangle \\ &\text{subject to} && y_i\big(\langle w, x_i \rangle + b\big) \geq 1 \end{aligned} \tag{2.14}$$

In order to allow for training instances to lie at the ‘wrong’ side of the decision boundary, per-point errors ξ_i known as *slack variables* are introduced in [14] as well:

$$\begin{aligned} &\xi_i > 1 && \text{misclassification of } i \\ &0 < \xi_i \leq 1 && \text{margin violation of } i \end{aligned} \tag{2.15}$$

Including the slack variables results in the following optimization problem [14]:

$$\begin{aligned} &\text{minimize} && C \sum \xi_i + \langle w, w \rangle \\ &\text{subject to} && y_i\big(\langle w, x_i \rangle + b\big) \geq 1 - \xi_i, \text{ with } \xi_i \geq 0 \end{aligned} \tag{2.16}$$

where C denotes how severely violations should be punished (e.g. $C = \infty$ turns definition 2.16 into 2.14). In [67] it is shown how the decision boundary can be moved to favour one class over the other by using different values of C for both classes, of which an application can be found in [4] where it is combined with under-sampling the minority class.

MetaCost provides a framework for applying misclassification costs for arbitrary classifiers based on a cost matrix consisting of misclassifications costs for every combination of classes C_{actual} and $C_{predicted}$ [16]: first, multiple copies of a classifiers are trained on different samples of the data. Predictions of these classifiers for all instances are aggregated by voting. Per-instance classification probabilities (i.e. the probability of an instance to be classified as class C in the vote-based classification) are thus defined as the fraction of votes for that class. Next, each instance is relabelled to the class that minimizes the result of the multiplication of the class probabilities with the misclassification cost for each class. Classes with a low misclassification cost (e.g. anomalous classes) can be expected to have more members when compared to original input. This can be thought of as a bias towards these classes. Finally, the classifiers are retrained on the relabelled set. Predictions for new instances are aggregated by voting as well.

Resampling sampling the minority and majority classes differently, so that the data on which the model is trained (the *training sample*) does not reflect the class-distribution in the original sample. There are two basic ways to achieve this [17]:

Oversampling minority instances of the minority class are duplicated in the training sample to increase the number of available examples. This can be advantageous when there is little data available and removing instances is undesirable. The duplication is sometimes extended to generation of new examples by adding noise [1].

Under-sampling majority not all instances of the majority class are included in training sample. This is preferred when enough data is present so that (ideally) only duplicate information is removed and the resulting training sample is smaller – resulting in faster training times.

The sampling probabilities can be set to reflect the costs of misclassification. Cost-based resampling is extremely similar to cost-sensitive classification. Under-sampling, however, may be beneficial to efficiency when representative samples can be constructed (i.e. enough data is present). A major advantage of resampling is that can be employed on top of any classifier. An extensive overview of resampling to correct for class imbalance can be found in [21].

Boosting training multiple simple (or *weak*) classifiers and aggregating their output to produce a final label can be beneficial when classes are imbalanced. Such a group of classifiers is generally known as an *ensemble*. AdaBoost from [23] is a classic example. In each round, a classifier is trained on a training set annotated with per-instance misclassification cost (the *weight*). These are adjusted, so that misclassified items receive heavier weights in the next round. The confidence or *strength* of the weak classifiers is decreased as a function of the number of previously misclassified items in a per-round fashion. The final label is determined by summing the strength-weighted outputs of all classifiers.

The above techniques can be applied both in 'generic' classification and anomaly detection. An important –often implicit– assumption they rely on, however, is that the original dataset contains a sufficiently representative set of outliers. This might pose a problem due to the limited availability of a representative set of anomalies. When data changes over time, new kinds of anomalies may enter the data set [26]. A convincing example of this can be found in anomaly detection for intrusion detection, where a model should be able to detect attacks of an entirely new *kind* as well as known attacks. A method that takes detecting unknown anomalies into account can be found in the next section.

2.2.2 Semi Supervised

Semi-supervised methods can be applied when some class labels are present and some are missing. Every supervised method can be 'transformed' into a semi-supervised method by completely ignoring the unlabelled instances. Although such an approach would lead to a method that adheres to the strict definition, only methods that actually take the unlabelled instances into account during training and classification are meant in most literature. In this section, some examples of semi-supervised anomaly detection methods are treated.

Similarly to the fully-supervised scenario, classification methods for the semi-supervised scenario can be adapted for the anomaly detection domain by weighting, resampling and boosting. A large variety of semi-supervised classification methods can be found in [71]. The two methods specifically designed for anomaly detection in [24] and [68] follow a different scheme: both rely on existing semi-supervised clustering techniques with an adapted objective function. The semi-supervised clustering methods they rely on are K-means clustering from [39] and fuzzy rough C-Means clustering from [31]. The objective function to minimize contains an overall clustering score, deviation from known instances and number of outliers. There are no test results of the algorithm proposed in [24]. The tests results in [68] indicate that results vary strongly on the chosen number of clusters, their precalculated centres and various other parameters. A method for obtaining these is part of future work. All of these semi-supervised methods originate from a supervised method and also require a representative sample of all anomalies in the training sample, which might be hard in some situations.

Another approach is taken in [26]. The authors argue that because of lacking information on possible (future) anomalies, it might be better to adapt unsupervised methods for semi-supervised learning instead of using generic semi-supervised classification methods as a basis. An SVM-inspired unsupervised technique from [60] by the name of *Support Vector Data Description* (SVDD) is used as a basis. This SVDD can be seen as an unsupervised one-class classifier: it forms a sphere around the data with radius R. The objective is to minimize the sphere, i.e. to minimize volume R^2 while still containing most of training examples. To enable non-spherical shapes, the problem can be solved in feature space by applying a kernel (see Section 2.1.2). Slack variables are used to allow for points to lie outside of the 'ball'. In the original paper, an adaptation that includes negative examples (i.e. labelled anomalies) in the optimization goal is described. It uses a similar approach as the weighted two-class SVM described in Section 2.2.1: all unlabelled points are assumed to be normal and the slack variables are weighted differently for wrongly classified anomalous and (assumed) normal points. However, in [26], it is shown that this problem is not guaranteed to be convex. This makes it possible for the solving algorithm to get stuck in local optima. Therefore, its authors propose an alternative formulation which is convex when combined with some types of kernel functions (most notably the popular *Radial Basis Function* (RBF) kernels), which they call Semi-Supervised Anomaly Detection (SSAD). This method outperforms existing methods on data sets in which new types of anomalies are introduced after training the model.

2.2.3 Active Learning

In this section we will treat some methods that can actively request labels for instances. Initially, all points are unlabelled. For a number of iterations, instances to be labelled are selected. Different criteria can be used for selecting these instances. The working model should be updated to reflect the feedback by the expert correctly. This setting is known as *Active Learning*. In active learning, the aim is to build some model while requesting as little labels as possible, i.e. to iteratively select unlabelled instances that would lead to the largest improvement of model performance. Generally, the expert is assumed to provide correct class labels all of the time.

The method in [47] assumes a mixture model fit to the data and its design is specific to be effective on large data sets with extremely few anomalies (0.001% outliers in a set of 10,000 instances). The method starts by clustering using the EM algorithm (see Section 2.1.1). Next, the following steps are performed in an iterative fashion: selecting the top-35 instances for labelling (1) and performing a semi-supervised alternative to EM (2). The following criteria are used alternately for selecting instances in step 1: instances which are far away from the clusters they have been assigned to by the EM algorithm and instances for which the likelihoods are similar for all components in the mixture according to the EM algorithm. These criteria are named *low likelihood* and *ambiguity* respectively the authors. Using these criteria in an alternating fashion yields the best results. The alternating of the low likelihood and ambiguity criteria is dubbed *interleave* and has proved successful in other settings as well [58]. The adaptation of the EM algorithm to include class labels for step 2 consists of overriding the class-probabilities to 1 for the actual class and 0 for all other classes after the E-step. This method will be referred to as 'Active Learning EM' in the remainder of this work.

Another active learning approach is presented in [69], where the unsupervised method LOCI (see Section 2.1.4) is combined with a two-class SVM. First, the MDEF values for all points for multiple radii r are computed. Next, all instances are sorted descending based on the maximum MDEF across all r. A number of suspected anomalies and normal points are selected from the top and bottom of this sorted list. Let POS and NEG denote the top and bottom of the list respectively. A two-class SVM is trained on the set, where POS and NEG form the training data. Next, the element closest to the margin on the negative (e.g. non-anomaly) side is selected. If the selected instance is an anomaly according to the expert, the subsequent next instance on the negative side of the decision boundary is selected. If the selected instance is not anomalous according to the expert, an instance is selected in between the last reported anomaly and the last reported normal instance. When no instances have been reported to be anomalous by the expert, an artificial instance on the decision boundary or the closest instance to the positive (i.e. anomalous) side can be used. The selection mechanism used here is generally known as the *margin strategy* for SVMs. It was originally presented in [63], where it was shown why it can be expected to lead to maximal classifier improvement in the general case. The feedback from the expert adds instances to POS and NEG, after which a new model is trained in the next iteration. Convergence is defined as receiving a negative answer on both type of queries to the expert. The robustness against the multi-granularity problem and against differences in local density and the SVMs guaranteed optimal separating hyperplane are presented as the methods main strengths.

The semi-supervised SSAD method presented in the previous section has been extended with a selection mechanism as well. As it is based on the unsupervised method SVDD, the first round does not have to be treated as a special case: the method bootstraps using vanilla SVDD. In subsequent rounds, points are queried by a combination of the margin strategy and a term that selects points that have little labelled neighbours. This latter term is used to maintain performance when new types of outliers are introduced. It consists of an adjacency matrix of training instances where all *verified* labels for each instance's k nearest neighbours are stored. By calculating the sum over all k nearest neighbours, clusters of previously unknown data points can be found. Points in these clusters will have a low value for this sum. Class labels for positive (normal) and negative (anomaly) classes are encoded as 1 and -1 respectively in SSAD, so that points which neighbour to both classes are also preferred in the next labelling round.

Chapter 3

Active learning for anomaly detection

This chapter contains a detailed description of the proposed method. The aim is for a method that satisfies the requirements described in Section 1.3, from which the following list of requirements was compiled . It is included for convenience and further reference:

1. **Presence of noise and anomalies**
   The method should be able to handle data sets in which both noise and anomalies are present besides normal entries. This is implied by sub goal 1 from Section 1.3.

2. **No class labels present initially**
   From sub goal 1 from Section 1.3, no class labels are present initially.

3. **Limited availability of domain expert**
   We assume a domain expert to have limited availability: this means that a domain expert is willing to label some entries, but would want to avoid having to label an entire data set. We are therefore interested in the trade-off between performance and the number of labelled instances.

4. **Reusability of result**
   An organisation can upload multiple data sets over time and will typically do so every month. Since we are dealing with HR data, we expect both the data and the definition of 'anomaly' to change only slightly within the time frame of a month. We aim to take advantage of this situation by using the outcome of a previous result for a next data upload. A model trained on a specific data set should be reusable on subsequent data uploads without losing adaptivity.

The first section of this chapter provides a high-level overview of the proposed framework: an unsupervised and a supervised component are combined with a selection mechanism to form an active learning method which outputs a classification to {anomaly, non-anomaly}. In subsequent sections we motivate the design decisions for the unsupervised and supervised components and the selection mechanism. We conclude this chapter with a description of the initial round, as it forms a special case.

3.1 High-Level Overview

The proposed framework consists of the following parts: an unsupervised component, a supervised component and a selection mechanism. For a graphical depiction, see Figure 3.1. Any unsupervised outlier detection that can generate an anomaly rank or score can be used for the unsupervised component. For a motivation of the anomaly rank or score requirement, see Section 3.4. The unsupervised method is necessitated by requirements one and two from the start of this chapter. The supervised component is necessary by the domain-specific nature of the separation between 'anomaly' and 'noise'. It can be any supervised method that produces an anomaly rank or score due to similar considerations as for the unsupervised method. Strictly speaking, any supervised method could be used, but in practice, methods that can handle class imbalance well are to be preferred. The selection mechanism is used to determine which points should be labelled next by the expert. These components are described in more detail below. For an overview of requirements and chosen methods, see Table 3.1.

From these separate component outputs however, a final anomaly score is still to be constructed. This is represented as a separate component in our framework, as multiple strategies can be considered. For example, a weighted average of the scores produced by both components could be used. The weight could be set once, determined as a function of the number of labelled exampled, etc. Since the only requirement for this component is that it can turn the outputs of the other components into a binary {anomaly, non-anomaly} classification, we refrain from describing it further in this chapter.

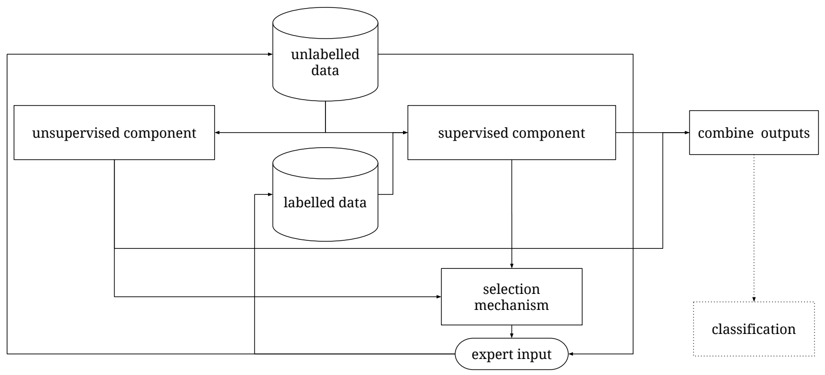


Figure 3.1: High-level overview of proposed framework. Unlabelled data forms input for a fully unsupervised component and a component with supervision. Any available labelled data forms additional input to the component with supervision. Together with a selection mechanism, the anomaly scores of these components lead to an instance to be labelled by the expert. Dotted lines indicate how the final binary classification is constructed after training.

	Fully Unsupervised methods	Fully Supervised methods (including C-SVC)	K-means clustering	Fuzzy Rough C-means clustering	SSAD	SVDD	Active Learning EM	LOCI and SVM
Presence of noise and anomalies (req. 1)		√	√	√	√	√	√	√
No class labels initially (req. 2)	√		√	√	√	√	√	√
Efficiently query labels (req. 3)					√	√	√	√
Robust against new classes of anomalies (req. 4)					√			
Anomaly score as output possible	√	√	√	√	√	√	√	√
Convex formulation to prevent convergence to local optima	N/A	N/A	N/A	N/A	√		N/A	√

Table 3.1: Comparison of methods from Chapter 2 based on requirements from the start of this chapter.

3.2 Unsupervised Component

As argued in Section 2.1, all unsupervised methods are based on implicit assumptions on characteristics of the data. The proximity methods from Section 2.1.4 are robust to differences in local density and make very little assumptions on data characteristics: they only assume a clustered shape and perform well on data with a lack of global correlations.

Because of above considerations, the LOF and LOCI methods seem most appropriate for the unsupervised component. Both are capable of returning per-point anomaly scores. For LOF, the original authors suggest to use a range for parameter k and select the maximum $\text{LOF}_k(i)$ of Equation 2.5 over all ks in the range for determining a final outlier score. LOCI defines outlier scores using an MDEF score over a range of radii r (see Equation 2.7) and provides a normalized standard deviation (see Equation 2.8) of these scores. We can convert these into an outlier score S_{LOCI} for each instance i in the following way [33]:

$$S_{\text{LOCI}}(i) = \underset{r \in \mathbb{R}}{max} \left\{ \frac{\text{MDEF}(i, \alpha, r)}{\sigma_{\text{MDEF}(i,\alpha,r)}} \right\} \tag{3.1}$$

We will compare these methods and select the method that gives most promising results.

3.3 Supervised Component

A supervised component is required to separate anomalies from non-anomalies (i.e. noise and normal points). This supervised component cannot consist of a fully-supervised method, as those methods require all instances to be labelled. From requirements two and three, we can deem this impossible: labelling an entire data set would be too costly and would not be an improvement over the current situation. The semi-supervised and active learning methods from Sections 2.2.2 and 2.2.3 do meet these requirements. From the remainder of this chapter, a supervised method that can handle missing labels (i.e. a semi-supervised or active learning method) is meant when describing the supervised component.

Requirement five from the start of this chapter implies that it is possible for new 'classes' of anomalous and normal points to be introduced after training via subsequent data uploads. This means that the supervised component should be robust to the introduction of new data points as well. The SSAD method uniquely combines robustness to introduction of previously unknown data with a semi-supervised approach and is therefore selected as a supervised component.

SSAD is rather novel and its performance characteristics have not been studied thoroughly. We therefore include a well-known state-of-the-art classifier that functions as a baseline for comparing performance. Since SSAD is a special kind of an SVM-based classifier, we select a more general SVM classifier (C-SVC) as a baseline for the supervised component.

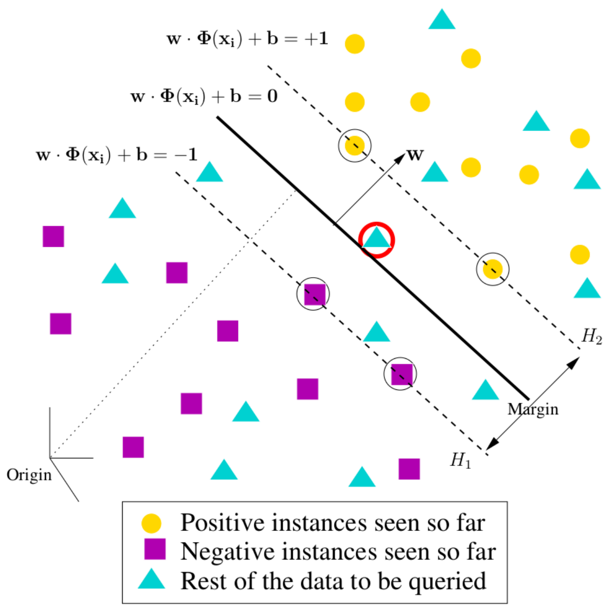

$\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i}) + \mathbf{b} = +\mathbf{1}$
$\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i}) + \mathbf{b} = \mathbf{0}$
$\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i}) + \mathbf{b} = -\mathbf{1}$
w
H_2
Margin
H_1
Origin
● Positive instances seen so far
■ Negative instances seen so far
▲ Rest of the data to be queried

Figure 3.2: Active Learning with SVM in a linearly separable example. The *version space* lies between the dashed lines marked by H_1 and H_2. The label for the unlabelled point closest to the margin (marked by red circle) is the most informative, as it is expected to halve the version space, irregardless of the actual label being positive or negative. Image taken from [19].

3.4 Selection Mechanism

The selection mechanism should enable the method to reach a good performance whilst requiring only little queries to the domain expert. It is the main driver for performance with respect to requirement three from the start of this chapter. In this section we propose a selection mechanism that combines *rank-based disagreement* with the *margin strategy*.

The margin strategy is an active-learning approach for SVMs. It was first presented in [63]. See Figure 3.2 for a graphical depiction. SVMs are binary classifiers that can find the optimal decision boundary between two classes. The solid line marked 'margin' forms the optimal decision boundary for the purple squares and the yellow dots in Figure 3.2. They do so by maximizing the distance from the decision boundary to labelled instances. The optimal decision boundary is one out of many separations that are consistent with the data. All consistent decision boundaries place all labelled instances at the correct side of the decision boundary, but the distance between the decision boundary and these points is not necessarily maximized. These possible separations lie in an area of feature space known as the *version space*. In Figure 3.2, the version space lies between the dashed lines marked with H_1 and H_2. In the semi-supervised scenario, unlabelled points about which the classifier is uncertain lie closer to the decision boundary than the labelled instances that helped calculating the optimal one. In [63] it is shown that the maximal classifier improvement can be achieved by dividing the version space into two equal parts in the uninformed scenario. This can be understood intuitively by considering that a query to the expert for the triangle encircled in Figure 3.2 can lead to a positive and negative label: a negative label will move the decision boundary roughly halfway towards the positive side whereas a positive label will move it roughly halfway away from the positive side. This strategy thus leads to the highest expected improvement to performance, given the current model and assuming no additional knowledge on the unclassified instances. Because of this it is often described as a 'greedy optimal' strategy.

The active learning extension to the proposed supervised component SSAD from [26] (see Section 2.2.3) therefore combines the *margin strategy* with a kNN-based heuristic in order to both investigate instances the classifier is uncertain about and focus on previously unseen parts of the data set respectively. This kNN-based strategy, however, implicitly assumes that all instances reside in equally dense parts of the data set – similarly to the kNN methods described in Section 2.1.4. We propose a method that is more robust to differences in local density by using using rank-based disagreement as a heuristic.

We assume both the supervised and unsupervised components can provide a per-instance *anomaly rank* and that the unsupervised component is robust to differences in local density. By comparing each instance's anomaly ranks from the supervised and unsupervised components, we calculate a metric of *disagreement* of both methods. By alternating queries for new labels using this *rank-based disagreement* and the margin strategy a similar scheme as the *interleave* method from [47] is achieved: instances about which the supervised method is uncertain and regions which appear to have not yet been properly included in training the supervised method will be selected in an alternating fashion. If methods A and B produce ranks $R_A(i)$ and $R_B(i)$, their rank-based disagreement $D(i)$ is denoted as:

$$D(i) = |R_A(i) - R_B(i)| \tag{3.2}$$

Note that both LOF and LOCI are robust to differences in local density and can generate anomaly ranks by ordering the instances based on outlier scores. The SSAD method can generate anomaly scores by using the distance to the decision boundary.

3.5 Initial Round

Because the supervised component needs *some* class labels, the very first round of learning forms a special case. The selection mechanism used in the initial round cannot be neither *disagreement* nor the margin strategy as both are based on a trained supervised model.

The solution is to fall back to a second unsupervised method in the initial round and start with the *disagreement* mechanism. Since the SSAD method is based on the unsupervised SVDD method, it is a natural fit to use this as the secondary unsupervised method in the initial round. For the baseline C-SVC method, there is no such fallback: we therefore set its performance to 0 until examples from both classes are available.

Chapter 4

Experimental setup

This chapter contains a description of the data sets and evaluation metrics used. The latter are derived from the problem statement in Section 1.3 and the requirements in Chapter 3.

We want to validate our method for the HR data use case, but still want to be able to compare it with other methods. We therefore include results on data sets that are well known in the anomaly detection literature. We first describe the data sets and how a ground truth has been established. We proceed with an extensive comparison on suitable metrics for determining the quality of anomaly detection methods. Next, we list the hyperparameters used and describe how their values were chosen. The chapter is closed by detailed descriptions of the experiments: the way generalisation was measured, how learning abilities and reusability were measured and which implementations were used.

4.1 Description of Data

Since we want to solve a real problem we want to measure performance on real data. We have access to an annotated relevant data set in the HR domain. We will denote this data set as the *domain data set*. This real-life data is not available for publishing as it contains confidential information.We therefore include modified versions of the *Abalone* and *Allhypo Thyroid* data sets.[1] These benchmark data sets are often used to reflect situations where a strong class imbalance is incorporated in the problem description [18] [28] [52]. They are both characterised by the presence of multiple minority classes and are often used to simulate unbalanced classification tasks. Note that we introduce the assumption of different 'classes' to be present given our the problem definition: it originally only mentions anomalies, noise and normal points. This 'transformation' of classification data to data for our problem statement is necessary, as there are no benchmark data sets tailored for our problem statement as far as we

[1]`https://archive.ics.uci.edu/ml/datasets.html`

know. An exact description of this transformation can be found in Section 4.2. We continue to describe the background of the benchmark data.

The *Abalone* data set originates from an original biological study [44]. The data was gathered with the goal of predicting the number of rings for specimen of a family of sea snails based on physiological measurements such as sex, length, height, weight, etc.

The *Allhypo Thyroid* data set was constructed as part of a study in the Machine Learning domain [53]. It describes patients suspected of having *Hypothyroidism*, a common disorder of the endocrine system in which the thyroid gland does not produce enough thyroid hormone. The data was gathered with the goal of predicting the presence and type of hypothyroidism based on attributes such as age, sex, and medical details such as the administered medicine, various measurements of hormone levels and (part of) the medical history of the patient. The data set contains individuals without hypothyroidism ('negative' class), individuals with hypothyroidism caused by a malfunctioning of the thyroid gland ('primary' class), individuals in an early stage of hypothyroidism in which the severity and completeness of symptoms are lower than for primary hypothyroidism ('compensated' class) and individuals with hypothyroidism with a specific cause ('secondary' class).

Categorical features were transformed into numerical features as 'dummy variables' for all data sets. For the supervised components all data was scaled to $[0, 1]$. The same scaling was used for training and test samples. No other normalisation was applied.

4.2 Establishing Ground Truth

The labels for the real life data set were obtained from a domain expert. Ideally, all entries would have been labelled, but this turned out to be too time consuming. Thus, all anomalies found in the original pre processing of the data as described in Section 1.1 were used the starter anomaly entries. In order to avoid misclassifying anomalies that were not found by the original pre processing, the labels for entries in the 10% of outlier scores for the LOF, LOCI and SSAD method were obtained in addition. This yields a total of 2.30% anomalies out of 1262 instances.

Making a distinction between anomalies and noise on the publicly available (Abalone and Allhypo Thyroid) data sets is problematic as no domain expert is available for this data. A common strategy to solve this problem is to use a data set with multiple minority classes and select one (or multiple) of these as anomalies. The Abalone data set is generally used for validation for classification and regression methods. We proceed by describing how this differentiation was made in this research.

The Abalone data set originally consists of 27 classes, which denote the number of rings found in observed abalone (no specimen with 28 rings in the sample). These were transformed into two classes as depicted in Figure 4.1: all entries from classes one up to five (1.83% of 2675 instances) were selected

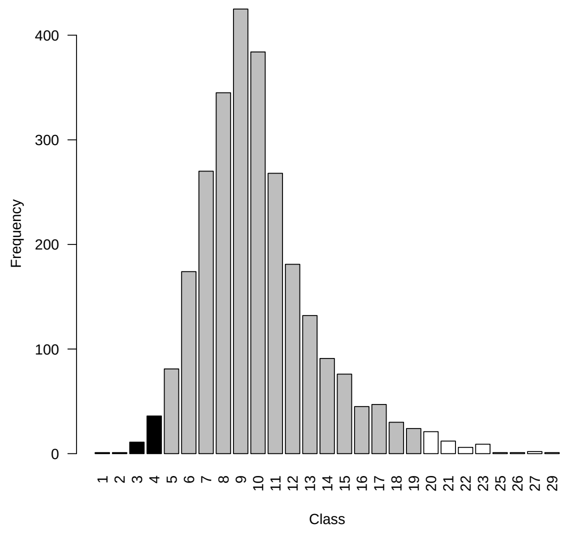


Figure 4.1: Histogram of occurrences of classes in the original Abalone data set. The four leftmost bars (black) form an artificial 'anomaly' class. All other classes are regarded as 'normal'. The data set contains minority classes at the rightmost tail as well (white), which can be regarded as the 'noise'.

to form an artificial anomaly class. The remaining classes (98.17%) form the non-anomalous cases. This latter set includes minority classes 20 to 29 with roughly the same number of instances as the anomalous class with 1.98% of the total data to represent the 'noise'.

For the Allhypo Thyroid data set, all entries containing missing data were removed. The remaining 'primary hypothyroid' (individuals with hypothyroidism caused by inadequate functioning of the thryoid gland itself, 2.57% of 1946) instances represent the anomalous class. The 'negative' class (individuals without hypothyroidism, 91.98%) represents the normal cases and the 'compensated hypothyroid' cases (individuals in which hypothyroidism is in a starting phase characterised by some of the symptoms of primary hypothyroidism, 5.45%) represent the 'noise'. The only instance with the 'secondary hypothyroid' was removed, resulting in a distribution that is visualised in Figure 4.2.

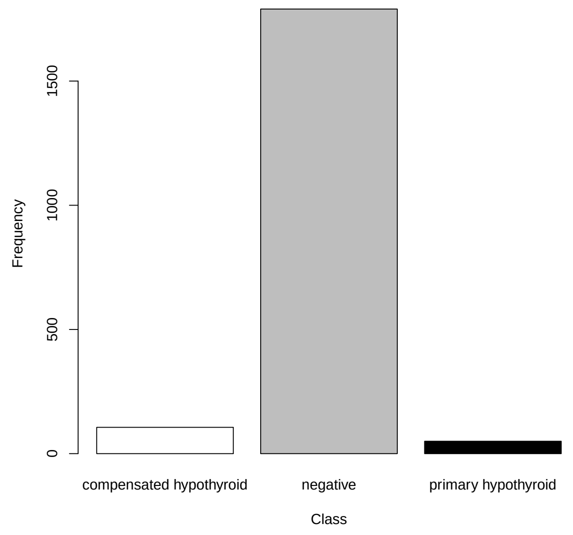


Figure 4.2: Histogram of occurrences of classes in the Allhypo Thyroid data set. The ‘compensated hypothyroid’ class (leftmost, white bar) forms the ‘noise’ class. The ‘negative’ class represents all ‘normal’ individuals. The ‘primary hypothyroid’ class (rightmost, black bar) forms the anomalous class.

4.3 Quality of Classifier

We have seen how the ‘ground truth’ of the class labels was established in Section 4.2. In this section, a way to compare the outputs of a classifier with this ground truth is described. As we have seen in Section 3.1, all components of the proposed approach output a per-instance anomaly score. Such a score can be converted into a binary classification by choosing a decision threshold θ: all instances with an anomaly score $> \theta$ are classified as outlier, all instances with an anomaly score $< \theta$ as normal. By varying the θ the performance metrics error rates can be influenced, as described into further detail below. We proceed by considering some classifier performance metrics for our anomaly detection task. For each method we note shortly how varying θ affects the result as measured by that metric. Finally, we argue that a metric called *AUC-PR* seems most suitable for comparing classifiers in the anomaly detection domain and that another metric named *F1 score* seems the most useful for assessing the performance of a combination of classifier and θ.

The (arguably) most simple evaluation metric for classifiers is *accuracy*:

$$Accuracy = \frac{TP + TN}{Total} \tag{4.1}$$

where TP, TN and $Total$ are defined in Table 4.1. The number of TP and TN can be influenced by adjusting the decision threshold θ: a more conservative θ is expected to increase TP whilst decreasing TN and vice versa. This metric is not

		actual		
		positive (+)	negative (−)	
predicted	positive (+)	*TP*	*FP*	*PP*
	negative (−)	*FN*	*TN*	*PN*
		AP	*AN*	*Total*

Table 4.1: Example confusion matrix. Positive predictions and actuals are denoted by a +, negatives by a −.

very informative for classifying outliers or anomalies, as the number of correct predictions might be very high if *every* point is classified as a normal point. Outliers and anomalies are rare by definition, which might lead to seemingly high accuracy when little anomalies are present.

Metrics that are able to take class imbalance into account should express the relation between the concepts in the confusion matrix in Table 4.1 better. Consider the following definitions:

$$Precision = \frac{TP}{PP} \qquad (4.2) \qquad\qquad \mathit{Recall} = \frac{TP}{AP} \qquad (4.3)$$

$$False\ Positive\ Rate = FPR = \frac{FP}{AN} \qquad (4.4)$$

which are more suitable in the case of class imbalance. They are used to construct two curves that visualise effects of adjusting θ: the *Receiver Operator Characteristics* curve (ROC curve) and the *Precision-Recall* curve (PR curve) which will be described next.

The ROC curve plots Recall (also known as True Positive Rate or TPR) against False Positive Rate (FPR). By doing so, a monotonic ascending curve towards $(1, 1)$ is formed. This facilitates comparisons between classifiers: the method or hyperparameter setting corresponding to the curve that dominates all other curves can be said to have superior overall performance. In addition, ROC curves aid in finding the right hyperparameter setting to achieve the desired Recall and FPR scores: since the ROC curve contains the full range of possible TPR and FPR scores, the trade off between these metrics by varying θ can be assessed at a glance. Generally, a liberal decision threshold θ leads to a high TPR at the cost of an increasing FPR.

As pointed out in [15], however, ROC curves are not useful in the presence of a strong class imbalance or when the types of errors should be weighted differently. Consider the hypothetical confusion matrices for two separate methods in Table 4.2. Next compare the derived scores for Precision, Recall and FPR:

	Method A		*Method B*	
Precision	$40/200 =$	0.2	$40/50 =$	0.8
Recall	$40/50 =$	0.8	$40/50 =$	0.8
FPR	$160/4950 \approx$	0.032323	$10/4950 \approx$	0.002020

The difference in performance of the methods is poorly represented in the FPR scores, as they are so small that any subtleties get lost. Comparing PR curves will better reflect differences between methods than comparing ROC curves when the number of false positives is expected to be relatively high. Making a distinction between a small set of entries and a large one is the goal in anomaly detection tasks, so we can expect this situation to be present often. Precision and Recall and their interaction are therefore more suitable for comparing classifiers in the anomaly detection domain. For Precision and Recall, a more liberal decision threshold θ is expected to result in a higher Recall at the cost of a lower Precision.

The method or hyperparameter setting that yields a PR curve that dominates the other curves can be viewed as having the best performance as it provides a more favourable trade-off between Precision and Recall. In order to convert PR curves into single-point scores, a measure of the area under the PR curve can be used (AUC-PR). We will therefore use AUC-PR when comparing classifier performance.

In order to determine how well a classifier ultimately performs the task of anomaly detection, however, the AUC-PR is not very useful as it describes performance of a classifier over an entire range of θ, whereas actual classifications can only be made after a specific decision threshold θ has been chosen. In order to select the optimal setting on the PR curve, a harmonic mean of Precision and Recall known as the *F1 score* can be used:

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.5}$$

		actual			actual		
		+	−		+	−	
prediction	+	40	160	200	40	10	50
	−	10	4790	4800	10	4950	4950
		50	4950	5000	50	4950	5000
		Method A			*Method B*		

Table 4.2: Example confusion matrices for hypothetical methods A and B on an imbalanced data set. Positive predictions and actuals are denoted by a $+$, negatives by a $-$.

In the following section we will describe how the AUC-PR and F1 score are used for hyperparameter selection.

4.4 Combining Component Outputs

The outputs of the unsupervised and supervised components can be turned into a final classification by various strategies. Example strategies are averaging, fixed weighted averaging and adaptive weighted averaging. In fixed weighted averaging, outputs are averaged using preset weights, whereas in adaptive weighted averaging the weights can be set as a function of e.g. the number of labelled training examples or based on hypothetical performance given the available labels. Weighted and adaptive weighted averaging prove promising, as the supervised component might outperform the unsupervised component when a sufficiently large labelled training set is present, whereas the unsupervised method can be expected to outperform the supervised method when there are insufficient labelled training examples. Expecting the supervised method to be able to leverage class labels effectively, we hypothesise that it will outperform the unsupervised component when a sufficient number of class labels is available. In order to limit the scope of this research, we therefore select a weighted average with weights set to 0 and 1 for the unsupervised and supervised components respectively, i.e. we let the supervised component be the sole contributor to the classification into {anomaly, non-anomaly}.

4.5 Hyperparameter Selection

The unsupervised LOF and LOCI and the supervised SSAD and C-SVC components all require some hyperparameters to be set. In order to evaluate the methods in a fair manner, the optimal hyperparameters have to be selected. This section describes how different hyperparameters affect classifier performance, how different hyperparameters were tested and lists the final selection of hyperparameter settings. We elaborate on the validity of using *optimal* hyperparameter settings and analyse the sensitivity of the methods involved in order to get an intuition on performance in the general case. We include some pointers on how parameters can be selected when the optimal settings are unknown.

4.5.1 Unsupervised Component

The unsupervised component never outputs an actual classification in our setting (see Section 4.4), but only ranks instances according to an outlier score that is used as input for the query selection mechanism. AUC-PR is sufficient in this situation and F1-score provides no relevant information since no actual classification decisions are made (see the previous section for details on these metrics). We are therefore only interested in AUC-PR scores for the unsupervised component. We will continue to describe which hyperparameters are required for LOF and LOCI and how the final hyperparameter setting was determined.

The original authors of the LOF method advise to calculate the $\text{LOF}_k(i)$ for a range of k for each instance i and select the highest outcome to determine an outlier score for i. The LOF method thus has two parameters k_{min} and k_{max} that denote the range of k. The lowest number of points to constitute a cluster should be represented by k_{min}. The upper limit k_{max} represents the number of points that may be in a cluster whilst still being regarded as outliers. The values for these hyperparameters can easily be set by the expert in some sensible way, e.g. as a percentage of the total number of instances in the data set. In this research, k_{min}, k_{max} yielding the best results according to a grid search in $[1, 200]$ for both k_{min} and k_{max} for each data set were used in further experiments. See Figure A.1 in Appendix A for a visualisation.

Judging from Figures A.1 and A.2 from Appendix A, the LOF method is not very sensitive to the k_{min} and k_{max} settings. The top-5 value pairs show PR-curves that are almost identical for all data sets and even considering the PR-curves for the top-1000 we see mostly equal curves. Although all further performance analyses in which the LOF method is included should be viewed as upper limits to performance, actual performance is not expected to degrade much when optimal k_{min} and k_{max} are unknown throughout this section.

For the LOCI method, a parameter r_{max} is used to denote the maximum range of instances to be in each other's counting neighbourhood $\mathcal{N}_{counting}$. The authors advise to set the value so that all points are always in each others $\mathcal{N}_{counting}$ for a precise result, i.e. by using the distance of the two points with the highest pairwise distance. As can be seen in Figures A.3 and A.4 in Appendix A, however, this value for r_{max} does not achieve the best results. The r_{max} yielding the best result was selected in order to make sure all methods are compared with equally 'optimized' hyperparameters.

Judging from Figures A.3 and A.4 from Appendix A, the LOCI method is somewhat sensitive to the r_{max} hyperparameter value. The top-5 values show PR-curves that are almost identical for all data sets, however, when considering the top-1000 we see strongly varying shapes. Performance analyses mentioned further below should therefore be interpreted as an upper limit on performance when LOCI is involved. Performance is expected to degrade in the typical case when the optimal r_{max} value is unknown.

4.5.2 Supervised Component

Both of the supervised components SSAD and C-SVC require two hyperparameters to be selected. Parameter C is a weight for penalizing slack variables. It thus represents the trade-off between complexity of the decision surface and the number of misclassified instances. A less liberal (i.e. higher valued) C leads to less errors on training data, a tighter fit of the enclosing hypersphere to the training data for SSAD and of the separating hyperplane between the classes for C-SVC. This can generally only be accomplished by a more 'complex' shape. A more liberal (i.e. lower valued) C allows for more errors on training data and thus a looser fit to the training data. The parameter C can thus be seen as a way to control over- and underfitting on the training data.

The second parameter is the kernel used. As SSAD poses a restriction on the used kernel due to the required convexity of the optimization problem we only use the *Radial Basis Function* (RBF) kernel. The RBF kernel has a parameter γ which denotes the peakiness of the feature space. A lower γ leads to a more smooth surface, whereas a higher γ leads to a more peaky surface. This parameter can thus be interpreted as an inverse of the 'influence' that a data point has (i.e. in being an anomaly or non-anomaly) and thus also influences the models' capability to 'generalise' from given input: a higher γ leads to a tighter fit to the training data and thus to more accuracy on the training set, whereas a lower γ leads to a looser fit and thus to less accuracy on the training set.

The optimal combination of hyperparameters was determined on a per-data set basis for both methods in the supervised component. The results can be found in Appendix B in Figures B.1 and B.2. Judging from the former, SSAD is rather sensitive to hyperparameter settings and no clear pattern can be identified. Generally, lower γs yield better performance than those in the higher ranges. The exception to this is the *Allhypo Thyroid* data set, which contains a lot of binary attributes that are converted to 0 and 1 during data transformation. It can be understood that this leads to lower distances than numerical attributes with higher ranges. The grid search for C-SVC in Figure B.2 shows a more clear pattern: the interplay between γ and C can be seen from the diagonal shape in the plots. The difference in optimal hyperparameter values over different data sets is clear here as well. The optimal values for hyperparameters γ and C are hard to determine as their interpretation is not straight-forward. All further performance analyses should thus be viewed as upper limits on performance. The performance for both methods is likely to be less in the case the optimal hyperparameter settings is unknown.

4.5.3 Parameter Settings

The hyperparameter settings in Table 4.3 were determined as described in the previous subsections as optimal on not-normalised data sets. They were used throughout further experiments. Please refer to the previous subsections for details and suggestions on determining these values.

	LOF		LOCI	SSAD		C-SVC	
	k_{min}	k_{max}	r_{max}	C	γ	C	γ
Domain data set	26	29	3900	1.08e-04	2.59e-02	7.28	2.40e-01
Abalone	91	91	0.8	5.74e-03	4.89e-03	1.27e-02	1.49
Allhypo Thyroid	198	199	33	78.80	7.88e-03	3.04e-01	853.17

Table 4.3: Optimal hyperparameter settings per data set.

4.6 Generalisability

Generalisability is the capability of a Machine Learning (M.L.) method when applied to a data set other than the set it was trained on (the *training set*). The other set is generally known as the *test set*. When an M.L. method performs well on unseen data, this indicates that it has captured patterns from the training set that are useful in the real world, rather than capturing incidental patterns, which is known as overfitting. Generalisability is an interesting metric in our use case, as it quantifies requirements 3 and 4 from Chapter 3: it expresses how well a previously trained model can be 'reused' without any input by the expert.

Regarding generalisability, we differentiate between two scenarios: the scenario where all data is available from the onset and the scenario where a previously trained model is applied to unseen data. The first scenario represents discovery of anomalies in a novel yet completely available data set, whereas the second scenario represents detection of anomalies for data sets that may change after a model has been trained. This latter scenario is interesting as all anomalies found in this scenario require no additional effort from the expert. We are interested in the capabilities of such a method to fully autonomously detect anomalies in new data sets, i.e. its *generalisability*.

In the first scenario, the method is trained and applied to the same data set. No cross-validation or other generalisability-measuring techniques are applied. In the second scenario, separate training and test sets represent the initial and new data sets respectively. Only the performance on the test set is relevant here. Five-fold cross validation was applied in these situations to measure the generalisability capabilities.A nested cross-validation was considered, however it was considered inappropriate due to too little positive (i.e. anomalous) validation instances in the inner cross-validation.

The unsupervised component typically has access to *all* data. Splitting data into 'train' and 'test' sets is therefore not necessary when only the unsupervised component is involved. For all runs in which only the supervised component is involved (i.e. hyperparameter selection, component performance analyses) a five-fold cross-validation was used to measure generalisation capabilities. To test the entire framework – e.g. a combination of supervised and unsupervised components, query and output combination mechanisms – performance metrics on both the test and training data were obtained.

4.7 Learning Abilities

In order to measure how our method performs with respect to limited availability of the domain expert, we want to compare the number of points labelled by the expert (the number of hints) with some performance metric. We will use the F1 score introduced in Section 4.3 and see how it is influenced by the number of labelled points by plotting it against the number of labelled points. This kind of plot is generally known as the *learning curve*.

When assessing the learning ability, we are interested in the learning abilities in the scenarios of a single and multiple data uploads. In the former, all data is present initially and the goal is to acquire all labels with as little queries to the expert as possible. For the learning abilities that span across data uploads we hold out part of the data set and use this only for validation.

4.8 Used Implementations

For the LOF method, an implementation from the 'DMwR' package of the statistical package *R* was used [64] [54]. An implementation from the 'ELKI' software package was used for the LOCI scores [56]. Scaling was performed using the 'LIBSVM' package [10]. The original implementation of the author was used for SSAD.[2] The C-SVC implementation from Scikit-learn was used [46]. Scikit-learn was also used for cross validation, parameter grid search, calculation of results (AUC-PR, F1 score, etc.) and the creation of plots.

[2] `https://github.com/nicococo/tilitools`

Chapter 5

Results

This chapter contains the results for the experiments described in the previous chapter. The results for the unsupervised and supervised components with optimal parameters are presented in detail. Next, the results of our proposed method for a varying number of hints by the expert are presented. A brief overview of our findings can be found first.

Local Outlier Factor (LOF) performs equally good or better than *Local Correlation Integral* (LOCI) for all data sets under consideration and was therefore selected as the unsupervised component. For the supervised component, the *C-Support Vector Classifier* (C-SVC) outperforms *Semi-supervised anomaly detection* (SSAD) for all data sets under consideration.

The proposed selection mechanism requires roughly 20% and 50% to reach maximum performance on the training and test data respectively, whereas the random selection mechanism requires all labels to achieve maximum performance. Performance on the training set, in which the labels for all instances can be queried, is better than on the test set, which is characterized by the absence of some data during training. Overall, incorporating user feedback only slightly increases performance when compared to a basic unsupervised approach, even when all labels are available. Expert feedback therefore remains valuable, even when a model was trained on an entire data set already.

5.1 Unsupervised component

The Precision-Recall (PR) curves from Figure 5.1 show the results of LOF and LOCI using per-dataset optimal parameters (see Sections 4.5.1 and 4.5.3 for details). LOF outperforms LOCI for the domain and Allhypo Thyroid data sets. For both data sets, the PR-curves for LOF dominate the LOCI curves almost completely: no matter what the preferred Precision-Recall trade-off is, LOF outperforms LOCI. The AUCs for LOF vs LOCI are 0.369 and 0.617 vs 0.158 and 0.390, i.e. the AUCs for LOF are 1.5 and 2.3 as much as the AUCs for LOCI for these data sets. The curve for the remaining Abalone data set resides

	LOF			LOCI		
	$max(F1)$	$\mu(F1)$	$\sigma(F1)$	$max(F1)$	$\mu(F1)$	$\sigma(F1)$
Abalone	0.209	0.082	0.048	**0.226**	0.067	0.051
Allhypo Thyroid	**0.637**	0.443	0.124	0.585	0.171	0.160
Domain	**0.511**	0.093	0.081	0.255	0.144	0.050

Table 5.1: F1 scores for LOF and LOCI on all data sets.

at the lower regions of the graph. The LOCI method performs only a little better on this data set with an AUC of 0.096, compared to an AUC of 0.089 for LOF, which is 1.07 as much. For this data set, however, both methods outperform each other for different Precision-Recall trade-offs. Considering that the AUCs are almost equal on this data set, performance could generally be considered as equal. Because of time constraints, we have chosen LOF as the preferred unsupervised component in further analyses. The significance of these results is further discussed in Section 6.5.

Sudden drops in Precision that do not yield any additional Recall (such as for at for LOF on the domain data set at a recall of little of 0.4) denote that the method is not capable of distinguishing amongst anomalies and non-anomalies for this part of its anomaly ranking: the rank by anomaly score matches the true rank of anomalies very poorly for these sections of the data set. The 'price' in Precision that is to be paid for additionally retrieved anomalies is relatively high at these drops, which leads to long sequences of normal points being reported as 'anomaly'. For example, the drop in performance for the LOCI method on the Allhypo Thyroid data set near the Recall of 7.5 would result in roughly 20 additionally misclassified anomalies, whilst no additional correctly classified anomalies are gained.

Consider the Precision values at a Recall of 1.0 for all curves: at a Recall of 1.0, all anomalies are correctly marked as such. For most method–data set combinations, Precision approaches 0.0, meaning that almost all points have to be labelled as anomaly in order to achieve a Recall of 1.0, the only exception being LOF on the Allhypo Thyroid data set.

There is a large difference in PR-curves for both methods on the domain data set: judging from the sharp decline in Precision at lower Recall-regions for LOCI, it incorrectly assigns some of its highest anomaly scores to non-anomalies. This contrasts with the performance for LOF, which only steeply declines at a recall of 0.4. It is furthermore interesting that the Precision for LOCI is better for a higher Recall on this data set.

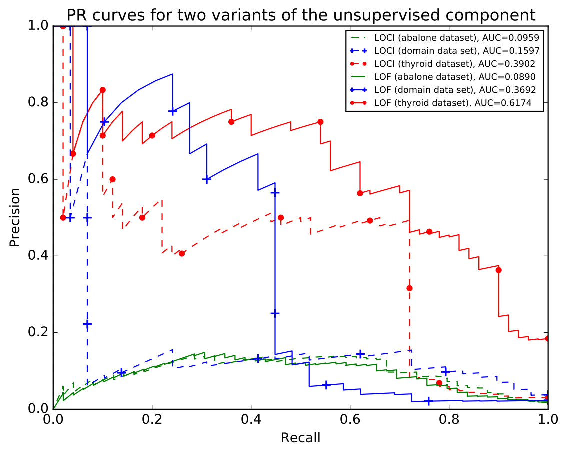


Figure 5.1: Precision-Recall curves for the unsupervised LOF and LOCI methods on all data sets under consideration.

5.2 Supervised component

The results using five-fold cross validation and per-dataset optimal parameters (see Sections 4.5.2 and 4.5.3 for details) for two supervised methods are listed in Table 5.2. The C-SVC method outperforms SSAD on all test data. Both of these methods are outperformed by methods with an unsupervised paradigm.

Note that the comparison between supervised and unsupervised methods is not based on equally sized test sets: the unsupervised scores are based on performance on the entire set, whereas the supervised scores are an average of F1 scores on the five test folds. We assume that any effects of this difference in test set sizes are accounted for by averaging the F1 scores for the five folds.

The unsupervised methods outperform the supervised methods on all data sets. This matches the rationale for SSAD, which is primarily based on an unsupervised method [26]. Interestingly, however, SSAD yields the lowest F1 scores in our experiments. In the original paper, SSAD outperforms traditional methods only when novel classes of anomalies are introduced, e.g. anomalies stemming from a different distribution. This could be an indication that in our scenario, the training and test samples stem from the same distribution, i.e. that there are sufficiently informative training examples available. Our experiments were not set up to reflect a situation of novel anomalies in the test set: we did not engineer our test set to contain anomalies from novel distributions as is done in the original paper. Although feasible, this explanation is not consistent with the results from C-SVC and the unsupervised methods. When we assume that the anomalies in the training set represent the anomalies in the test set well, we would expect C-SVC to be able to outperform the uninformed unsupervised methods, since the C-SVC method is theoretically able to differentiate between anomalies and outliers, whereas an unsupervised method is not.

The inconsistency introduced by the unsupervised methods outperforming

	Supervised		Unsupervised	
	SSAD	C-SVC	LOCI	LOF
Abalone	0.0	**0.038**	0.226	0.209
Allhypo Thyroid	0.019	**0.136**	0.585	0.637
Domain	0.124	**0.489**	0.255	0.511

Table 5.2: F1 scores for all tested methods on all data sets on the test set (supervised methods) and training set (unupservised methods). The supervised methods were tested using five-fold cross validation.

C-SVC under the assumption that the training data represents the test data well can have various reasons. We identify two independent causes of this inconsistency in this paragraph. Firstly, we optimized F1 score for the unsupervised methods as we can influence θ directly, but did not do so for SSAD and C-SVC as we depend on these methods to find the optimal separating hyperplane themselves. We therefore have to use γ and C to tweak performance, which are known to be hard to set and might influence each other indirectly. A comparison between unsupervised and supervised methods is not representative for actual performance in this regard. The $\mu(F1)$ and $\sigma(F1)$ scores indicate that the $max(F1)$ scores are not representative throughout the entire PR curve. However, the unsupervised methods (LOF especially) appear to be able to perform well for non-optimal parameters as well from the figures in Appendix A: the PR curves yielding the top-5 and top-1000 are not very different. Quantifying the effects of the parameters for these methods is left for future work. Another possibility is that SSAD and C-SVC are not capable of capturing the complexity of the underlying relations well enough. This is possibly the case: both LOF and LOCI by their definition use information on the local density of points to construct anomaly scores. Local density is not explicitly encoded as input to the supervised methods and thus part of 'being an anomaly' might not be captured well by only confining regions of the data domain.

5.3 Learning Abilities

From Section 5.1 we have seen the maximal performance for an unsupervised method. In this section we examine how well our proposed method performs in comparison. We therefore examine F1 score, Precision and Recall for different numbers of labels on a training set, i.e. all data is available throughout the entire method. Furthermore, we are interested in the effects of using our query mechanism over a random query, which serves as a lower-bound on performance without our proposed method. Because of time considerations, we only include results for a combination of the most promising supervised and unsupervised methods, e.g. LOF and C-SVC. Note that all data is available during query selection but not during training of C-SVC for final classification. Only data for which labels are available are used during this phase, as C-SVC is not capable

	Proposed method	Unsupervised methods
Abalone	0.056	**0.226**
Allhypo Thyroid	**0.681**	0.637
Domain	**0.609**	0.511

Table 5.3: Comparison of maximum F1 scores on training data for the proposed method and for unsupervised methods. Bold denotes per-row maximum.

of taking instances with missing labels into account.

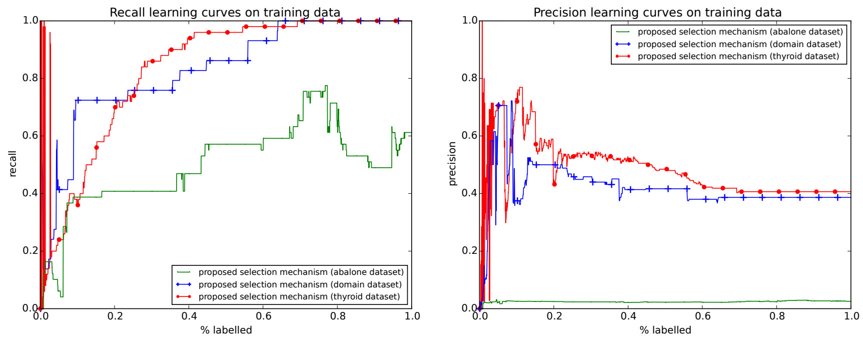
Judging from Table 5.3, only a slight F1 score increase over the unsupervised methods is achieved when using the available labels for the Allhypo Thyroid and domain data sets. The proposed method is incapable of outperforming the unsupervised methods for the Abalone data set. Further investigating performance differences by comparing Precision and Recall trade-offs in Figures 5.1 and 5.2 in Table 5.4, our proposed method does show a number of beneficial characteristics. It achieves a Recall of 1.0 with reasonable Precision for the domain and Allhypo Thyroid data sets (Table 5.4). Especially for the domain data set, on which the unsupervised methods fail to achieve a practical Precision at a Recall > 0.5, the benefits of using the acquired labels are noteworthy: a Recall of 0.5 is achieved at less than 20% of labelled instances with a Precision > 0.4 (Figures 5.1 and 5.2). Regarding performance on the training set, our method mainly outperforms the unsupervised methods whenever a high Recall is required and only when a sufficient amount of labels is available. The significance of these results is discussed in Section 6.5.

It is noteworthy that an F1 score of 1.0 is not achieved for any data set – even when all labels are available during training. This indicates that our classifier is unable to define a good separation between the classes in the training examples. For C-SVC, it means a separating hyperplane in which all instances lie at the correct side could not be computed. This situation is known as the 'non separable case'. Assuming that our model can express sufficient complexity, non-separability indicates that out of multiple identical instances, some are reported as anomalies whereas others as non-anomalies. By using appropriate hyperparameters, F1 scores of 1.0 on the training set were obtained for all data sets, indicating that there are no identical instances that belong to different classes.

In a worst-case situation, the expert inspects instances in an arbitrary order, thus labelling them randomly. In order to gain insight in how our informed query mechanism affects performance, we compare the dashed and solid lines in Figure 5.3 and we compare the maximum achieved F1 scores in Table 5.5. We clearly see the benefits from using an informed query mechanism over random querying. The learning curves for the proposed query mechanism dominate the random query mechanism for the domain and Allhypo Thyroid data sets. The proposed query mechanism yields higher maximum F1 scores and requires less labels to do so. The performance on the abalone data sets is comparable amongst the query mechanisms.

	Proposed method			Unsupervised	
	$max(Recall)$	at Precision	at % labelled	Precision	Method
Abalone	0.775	0.026	69.3%	**0.031**	LOCI
Allhypo Thyroid	1.0	**0.403**	78.2%	0.184	LOF
Domain	1.0	**0.367**	63.9%	0.038	LOF

Table 5.4: Comparison of Precision scores at maximum Recall on training data in proposed method and in unsupervised method. When maximum Recall is achieved for various 'at % labelled' and thus various corresponding Precision scores, the lowest 'at % labelled' is shown as further labelling would not uncover additional anomalies. Note that the oscillations at the beginning of the graph have been omitted, as they include models that 'cheat' by classifying everything as anomaly. Bold denotes per-row maximum Precision.



(a) Learning curve based on Recall. (b) Learning curve based on Precision.

Figure 5.2: Learning curves based on Precision and Recall on the training set.

	Proposed selection		Random selection	
	$max(F1)$	at % labelled	$max(F1)$	at % labelled
Abalone	**0.056**	88%	0.052	90%
Allhypo Thyroid	**0.681**	41%	0.579	100%
Domain	**0.609**	13%	0.557	100%

Table 5.5: F1 scores for the proposed selection strategy and a random selection on all data sets on the training set. LOF and C-SVC were used as the unsupervised and supervised component.

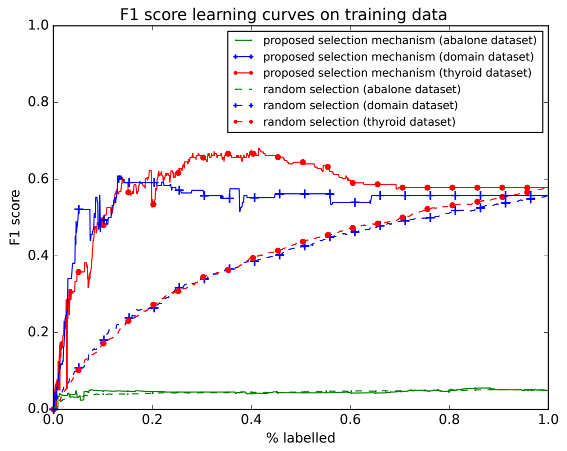


Figure 5.3: Learning curves for the proposed method on all data sets on the training set. The 'random selection mechanism' is included for comparison. For the 'random selection' results, an average of 25 runs is used to remove the effects of incidents.

5.4 Generalisability

Knowing how many labels we need to query in order to achieve a desired performance on training data, we do not yet know how well our proposed method will perform on test data. Recall that our model is intended to be used in a setting where multiple data uploads are possible and where it would be possible to apply a previously trained model on new data sets. We do this by looking at the generalisability of our proposed model, e.g. by investigating performance on a separate test data through five-fold cross validation. During training, not all data can be requested by our proposed query mechanism. The effects of this limitation in choice for labelling can be found in this section.

The differences between the scenarios are clear when comparing Figure 5.3 with Figure 5.4 and Table 5.5 with Table 5.6. All methods perform generally better on training data. The difference in performance on the Allhypo Thyroid data set is bigger than the difference for the domain data set. Similarly to the results listed above, the performance on the Abalone dataset is very different from the other two data sets.

Even though performance degrades when not all instances are known up front, the proposed selection mechanism still increases performance and requires less labelled instances to reach the optimal results (from 67% up to 9% of total number of instances).

The learning curves based on Recall and Precision from Figure 5.5 contain the same oscillations as those in Figure 5.2 in the beginning stage. The Recall reaches maxima of roughly 0.6 for the abalone and domain and 0.2 for the thyroid dataset. Precision at these points is nearing 0.0 (abalone), 0.6 (domain) and 0.1 (thyroid). Furthermore, we note that the Recall for the Allhypo Thyroid

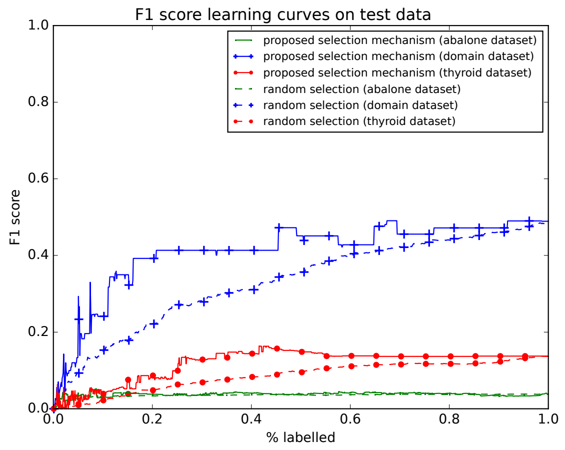
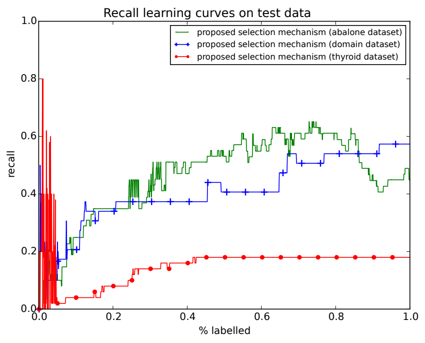


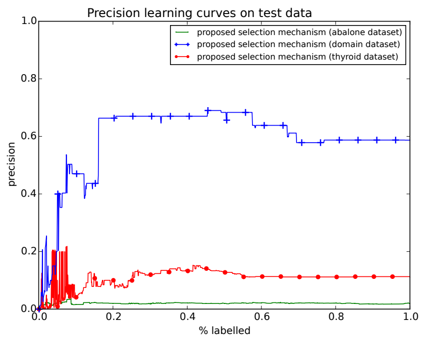
Figure 5.4: Learning curves for the proposed and a random selection mechanism on test data. A 'random selection mechanism' is included for comparison. For the 'random selection' results, an average of 25 runs is used to remove the effects of incidents.

	Proposed selection		Random selection	
	$max(F1)$	at % labelled	$max(F1)$	at % labelled
Abalone	**0.052**	9%	0.038	95%
Allhypo Thyroid	**0.164**	42%	0.137	99%
Domain	**0.491**	67%	0.489	100%

Table 5.6: F1 scores for LOF and LOCI on test data. LOF and C-SVC were used as the unsupervised and supervised component.



(a) Learning curve based on Recall.



(b) Learning curve based on Precision.

Figure 5.5: Learning curves based on Precision and Recall on test data.

data set does not improve after having labelled little over 40% of the data set. This might be caused by the lack of additionally found anomalies after this point and/or because of a lack of anomalies representative to those in the validation set during training.

Chapter 6

Discussion

In this chapter, we reflect on the choices made in this research and investigate key assumptions and their implications. First, we treat a contradiction in the requirement of being able to handle a lack of *a priori* knowledge and the varying definitions of outliers. Secondly, we investigate the contribution of this research to existing work. We continue this chapter by a discussion on some of the results in order to enlarge the comprehension of the problem and methods used in this chapter, including some suggestions for further research. This chapter is closed by some remarks on the usage of the proposed method in a real-world scenario.

6.1 Lacking *a priori* Knowledge

The task of detecting outliers without *a priori* knowledge requires some closer inspection. From the analysis of related work in Chapter 2, we have seen how a taxonomy of methods can be built from different definitions of the term outlier. It can be understood that different definitions of the term outlier are expected to lead to different performance on a given data set: when the definition of outlier that is encoded into the data set by the actual class labels or by judgement of the expert matches the definition that drives the method being applied to the data, we can expect performance to be optimal. Consequently, there is a relation between the choice for unsupervised outlier detection methods and up-front knowledge on the data or problem domain. We proceed to investigate how this relation of unsupervised methods with predefined definitions of 'outlier' affects our 'no *a priori* knowledge' requirement.

An up-front notion of what an outlier looks like, is of course a form of *a priori* knowledge. Any choice for an unsupervised method should come from some notion of outlier, which conflicts with our goal of finding outliers without *a priori* knowledge. This contradiction could be seen as the end point for any research in this direction. A further inspection of the various unsupervised methods, however, shows that the various definitions of outliers make assumptions that are of a different nature and of a different strictness. Some methods, for example, require an assumption on the underlying distribution of the data or on the presence of globally consistent correlations, whereas others have less far-reaching assumptions such as data being clustered in some form or a distance metric being applicable to the data set.

We have attempted to go on with our research by selecting unsupervised methods with relatively narrow assumptions and robustness against specific characteristics of the data set, by using density-based methods with robustness against differences in local density. Although this alleviates the problems with a lack of up-front knowledge somewhat, it does not solve these problems completely. Some further research in this direction could be to find a definition of outlier that matches definition of the expert more explicitly. An alternative approach to solving the problem in this research is to reformulate it as learning the correct outlier definitions based on user feedback. A simple approach would be to create multiple models simultaneously and select the one with the best performance. Labels could be queried via the *ambiguity* and *low likelihood* strategy as in [47]. A more sophisticated approach would consist of applying user feedback in finding the best combination of multiple unsupervised models, a technique that is known as *ensemble learning*. The former suggestion is more straightforward than the latter, whereas the latter is more flexible. The latter suggestion is specifically interesting in the light of the requirement of lacking *a priori* knowledge, as it also does not require the assumption of a single definition of 'outlier' that matches all of the data well.

6.2 Novelty of Proposed Method

The notion of separating noise and anomalies through a combination of unsupervised and supervised methods as presented in this research is uncommon in this domain. The lack of an expert that is willing and accessible to provide correct labels might be one of the causes for this. We note that in our use case, such an expert is not only accessible, but would have to do the labelling regardless of our method being in place. The costs that are introduced by adopting the proposed method would therefore fully consist of training the model during data set validation once a robust implementation has been made.

The novelty in the approach taken lies in iteratively training a (possibly fully) supervised component by feeding a selection mechanism with the results from both a supervised and an unsupervised model. Similar attempts either use an unsupervised model only once or contain a single semi-supervised model to achieve 'active learning' [69] [63] [47]. We believe this contribution could be extended by populating the unsupervised component by an ensemble of unsupervised models. This opens the direction for finding different *kinds* of outliers efficiently, from which the supervised component might benefit heavily. Another extension would be incorporating outlier scores produced by the unsupervised component in the final classification. Both of these suggestions are possible directions for further research.

6.3 Limitations of Selection Mechanism

Our proposed selection mechanism is based on (1) the rank-based disagreement of the supervised and unsupervised component and (2) the margin strategy. We derive anomaly ranks for the supervised component from the distances of instances to the decision boundary. Since the margin strategy is also based on distance to the decision boundary (see Section 3.4 for details), both elements of the proposed selection mechanism are based on the same heuristic. This might result in some regions of the data set not being investigated properly. When instances in a region are far from the decision boundary, we fully depend on the unsupervised method to flag these these points as possible outliers. We propose two extensions to our method that could be investigated in future work.

The first proposed extension relies on employing a *stochastic* margin strategy in which a selection probability is assigned to each instance; the actual selection is by random selection based on the selection probabilities. The selection probabilities can be determined as a function of the distance to the decision boundary. The most straightforward approach would sum all distances and give each instance part of a roulette wheel that matches its distance as a proportion of this sum. This 'roulette wheel' approach can be modified into more elaborate schemes, such as ranking all instances based on distance and sampling them according to some distribution based on the order. For example, sampling from the ranked instances by the exponential distribution would lead to the closest instances having a high chance of being selected.

Another more direct strategy to counteract the problem of only querying instances in a specific region of the data domain can be found in [26]. An *adjacency matrix* in which instances that are close-by is built up. Labelled instances propagate their labels to their k-nearest neighbours as 'dummy labels'. Regions of the data that have not been inspected by the expert can be found by looking for clusters that have no or little labels after propagation. Extending this adjacency matrix to include robustness against differences in local density is to be investigated in future work.

6.4 Generation of Labels

As noted in Section 4.2, not all instances in the Domain data set were labelled by the expert. An original labelling as done by the expert was extended by a second labelling round in which instances which flagged as outlier from various methods (LOF, LOCI, SSAD) with various parameter settings. This way of establishing ground truth has introduced some bias towards these methods in this research. It is unclear how many instances were originally investigated, but 10 instances were flagged as anomaly prior to this research. The remaining 19 anomalies were found after labelling about 52 instances elicited by anomaly detection methods. The bias introduced by this practice could be significant, which is subject to further investigation. Note that this bias is only present for the domain data set.

6.5 A Note on Generalisability

As we have seen from the Recall learning curve in Figure 5.2a the method can aid in finding anomalies in a single data set more quickly. It does not require any class labels to be present initially. The model, however, fails to find a satisfactory number of anomalies on unseen data (Figure 5.5a). A simple unsupervised method (e.g. LOF) outperforms our proposed method on test data in terms of its optimal F1 score – taking into account that the unsupservised methods were optimized on F1 score directly (e.g. by influencing decision threshold θ) and the supervised methods were optimized on unseen data indirectly (e.g. by influencing model hyperparameter C and γ). Incorporating the output of the unsupervised method as suggested previously in this chapter could boost performance here too as apparently the unsupervised method can elicit outliers on previously unseen data better than the trained supervised models. In addition to this possibility for further research, we advise to always put some of the previously unseen data up for labelling to the expert and not rely on a model that was trained exclusively on a single data upload.

We did not investigate whether the differences between performance of the unsupervised and supervised methods are statistically significant. Some of the differences appear significant, e.g. the $max(F1)$ for LOF is slightly over two times as much as its LOCI counterpart on the Domain data set in Table 5.1, whereas other differences may be attributable to noise, e.g. the comparison of performances on the Allhypo Thyroid data set in Table 5.3. In order to get a good understanding of how well our method generalises, and whether it actually adds anything to a bare unsupervised methods, a statistical analysis of these differences is required.

Chapter 7

Conclusion

We have presented a framework for efficiently finding interesting outliers in data sets about which little prior knowledge. We specifically treat the scenario where an expert is available to provide labels as requested by the framework. The framework can be described as an active learning approach to anomaly detection and consists of an unsupervised and a supervised component with a query selection mechanism.

In order to detect outliers efficiently without *a priori* knowledge, an unsupervised method that makes little assumptions on the generating process or structure of the data is favourable. After examination of several models for outlier detection we deem the density-based model to be favourable for this use case: it poses little assumptions on the data and is robust against differences in local density. We found Local Outlier Factor to be the most suitable in this class of outlier detection methods.

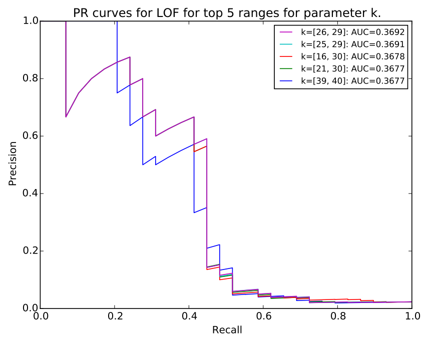
We have identified supervised anomaly detection as a variant to regular classification, characterised by a strong class imbalance. We have proposed an active learning approach to separate anomalies from noise and normal points. The hypothesis that a semi-supervised model based on an unsupervised SVM-like model would outperform a more traditional supervised multi-class SVM was not supported by results. The traditional multi-class SVM outperformed the specialised SSAD method in all experiments.

We have shown a way of effectively using expert input by querying labels in an informed way. The proposed selection mechanism alternates querying for points that are suspected by the unsupervised component with querying for points about which the actual classifier is uncertain. Final classification performance was increased by this method both when all data was available throughout training and when new data was introduced. This query mechanism even enables classification performance to exceed the performance when all labels are present present with roughly 20% of the labels queried.

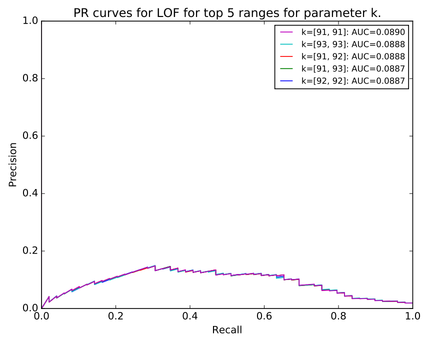
Our proposed method has shown a performance gain when all off the data is accessible for labelling in the training phase. In such a scenario, roughly 75% of all anomalies were found at a precision of 0.5. The necessity of an unsupervised method when applying data was shown from the detrimental performance on applying a trained model on wholly unseen data. Using a supervised model to select a (combination of) suitable unsupervised method(s) based on user feedback is part of future work.

Appendix A

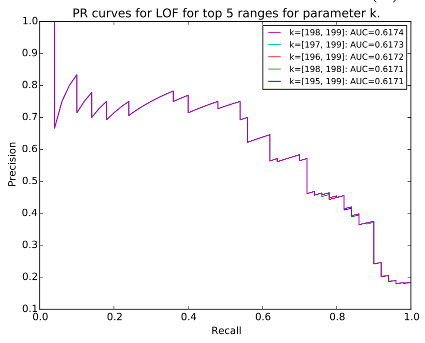
Hyperparameter Selection Unsupervised Component



(a) Domain dataset

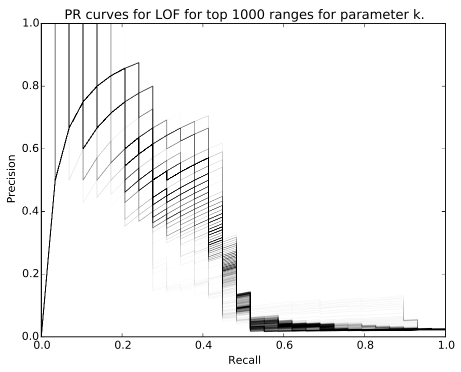


(b) Abalone

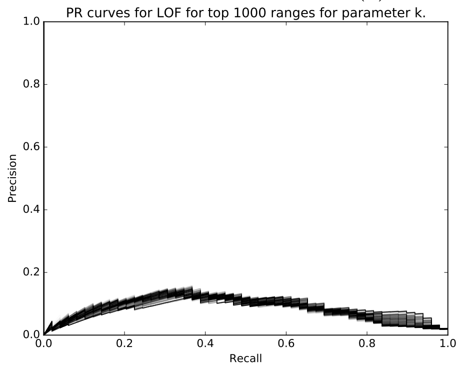


(c) Allhypo Thyroid

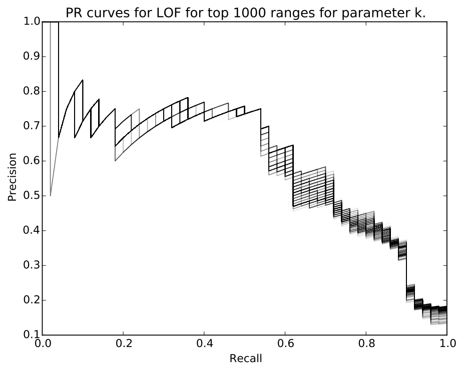
Figure A.1: The PR curves for top-5 settings of k_{min}, k_{max} for LOF on all used datasets, based on AUC-PR.



(a) Domain dataset

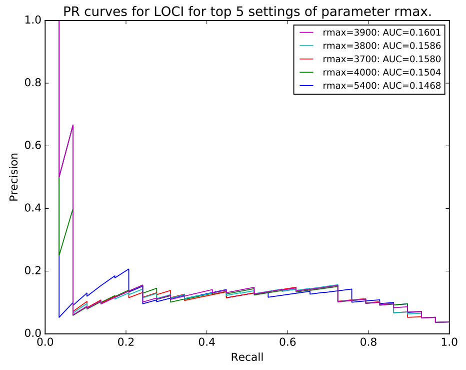


(b) Abalone

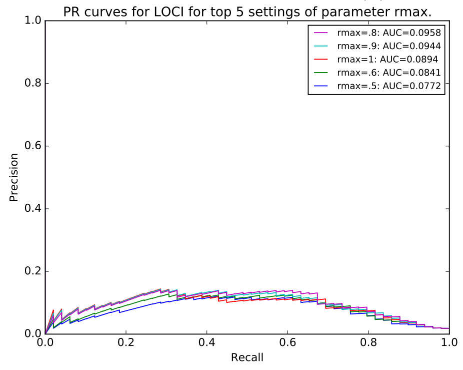


(c) Allhypo Thyroid

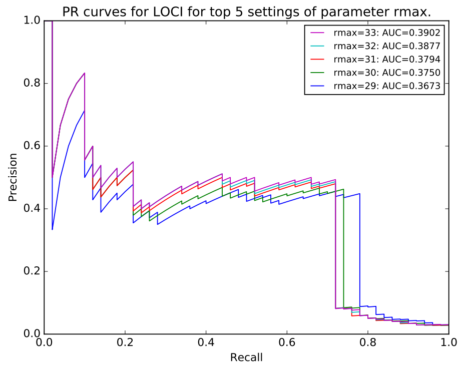
Figure A.2: The PR curves for varying values of parameters k_{min}, k_{max} for LOF on all used datasets.



(a) Domain dataset

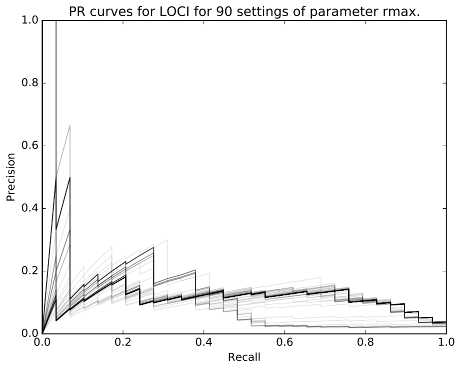


(b) Abalone

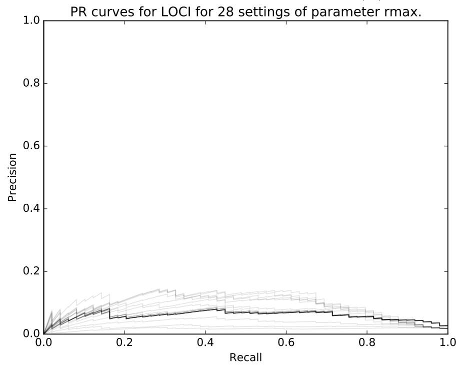


(c) Allhypo Thyroid

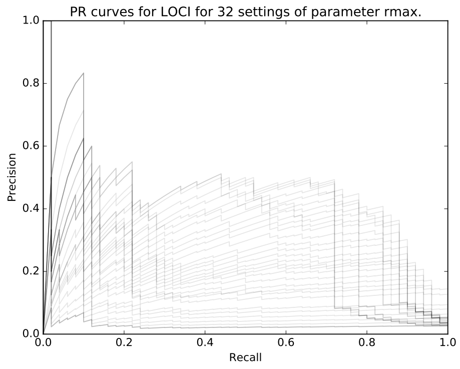
Figure A.3: The PR curves for top-5 settings of r_{max} for LOCI on all used datasets, based on AUC-PR.



(a) Domain dataset



(b) Abalone

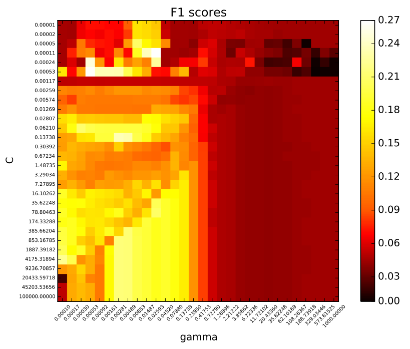


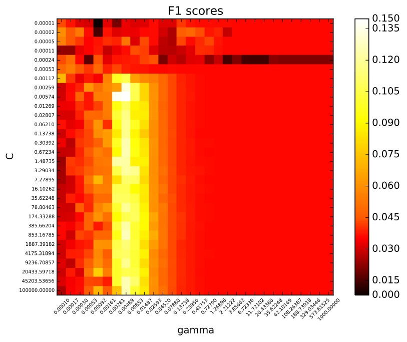
(c) Allhypo Thyroid

Figure A.4: The PR curves for varying values of parameters r_{max} for LOCI on all used datasets.

Appendix B

Hyperparameter Selection Supervised Component


(a) Domain dataset


(b) Abalone

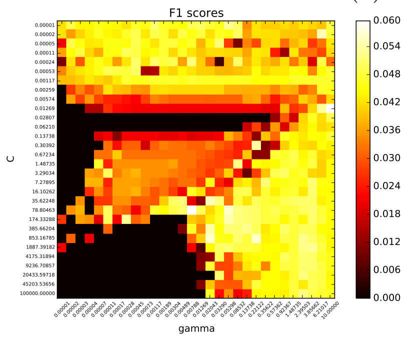
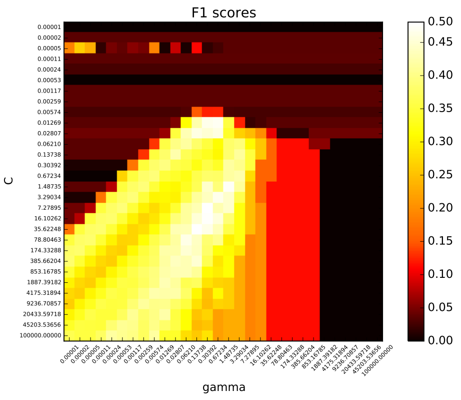
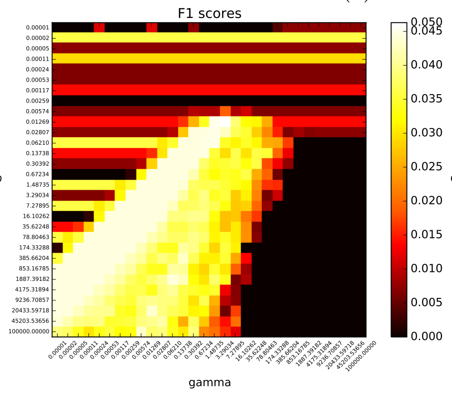

(c) Allhypo Thyroid

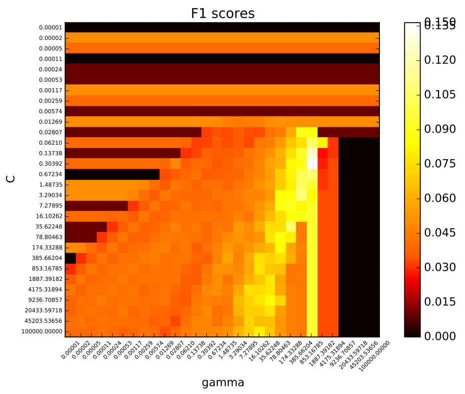
Figure B.1: Results of five-fold cross validated parameter grid search for SSAD method on a logarithmic scale for parameters C and γ.



(a) Domain dataset



(b) Abalone



(c) Allhypo Thyroid

Figure B.2: Results of five-fold cross validated parameter grid search for C-SVC method on a logarithmic scale for parameters C and γ.

Bibliography

[1] Naoki Abe, Bianca Zadrozny, and John Langford. Outlier detection by active learning. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 504–509. ACM, 2006.

[2] Charu C Aggarwal. *Outlier analysis*. Springer Science & Business Media, 2013.

[3] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. *On the surprising behavior of distance metrics in high dimensional space*. Springer, 2001.

[4] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *Machine Learning: ECML 2004*, pages 39–50. Springer, 2004.

[5] Mennatallah Amer, Markus Goldstein, and Slim Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pages 8–15. ACM, 2013.

[6] Vic Barnett and Toby Lewis. *Outliers in statistical data*, volume 3. Wiley New York, 1994.

[7] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[8] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[9] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM Sigmod Record*, volume 29, pages 93–104. ACM, 2000.

[10] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[11] William Chavuenet. *A manual of spherical and practical astronomy.* 1871.

[12] Chao Chen, Andy Liaw, and Leo Breiman. Using random forest to learn imbalanced data. *University of California, Berkeley*, 2004.

[13] Yixin Chen, Xin Dang, Hanxiang Peng, and Henry L Bart. Outlier detection with the kernelized spatial depth function. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):288–305, 2009.

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[15] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[16] Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164. ACM, 1999.

[17] Chris Drummond, Robert C Holte, et al. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11. Citeseer, 2003.

[18] Andrew F Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. Systematic construction of anomaly detection benchmarks from real data. In *Proceedings of the ACM SIGKDD workshop on outlier detection and description*, pages 16–21. ACM, 2013.

[19] Seyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. Learning on the border: active learning in imbalanced data classification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 127–136. ACM, 2007.

[20] H Jair Escalante. A comparison of outlier detection algorithms for machine learning. In *Proceedings of the International Conference on Communications in Computing*, pages 228–237. Citeseer, 2005.

[21] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

[22] Peter Filzmoser, Robert G Garrett, and Clemens Reimann. Multivariate outlier detection in exploration geochemistry. *Computers & Geosciences*, 31(5):579–587, 2005.

[23] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[24] Jing Gao, Haibin Cheng, and Pang-Ning Tan. Semi-supervised outlier detection. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 635–636. ACM, 2006.

[25] Robert G Garrett. The chi-square plot: a tool for multivariate outlier recognition. *Journal of Geochemical Exploration*, 32(1):319–341, 1989.

[26] Nico Görnitz, Marius Micha Kloft, Konrad Rieck, and Ulf Brefeld. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research*, 2013.

[27] Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

[28] Hongyu Guo and Herna L Viktor. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *ACM SIGKDD Explorations Newsletter*, 6(1):30–39, 2004.

[29] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[30] Victoria J Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[31] Qinghua Hu and Daren Yu. An improved clustering algorithm for information granulation. In *Fuzzy Systems and Knowledge Discovery*, pages 494–504. Springer, 2005.

[32] J Janssens and E Postma. One-class classification with lof and loci: An empirical comparison. In *Proceedings of the Eighteenth Annual Belgian-Dutch Conference on Machine Learning*, pages 56–64, 2009.

[33] JHM Janssens. *Outlier selection and one-class classification.* PhD thesis, Tilburg University, 2013.

[34] Richard Arnold Johnson, Dean W Wichern, et al. *Applied multivariate statistical analysis*, volume 4. Prentice hall Englewood Cliffs, NJ, 1992.

[35] Theodore Johnson, Ivy Kwok, and Raymond T Ng. Fast computation of 2-dimensional depth contours. In *KDD*, pages 224–228. Citeseer, 1998.

[36] Edwin M Knox and Raymond T Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the International Conference on Very Large Data Bases*, pages 392–403. Citeseer, 1998.

[37] Hans-Peter Kriegel, Arthur Zimek, et al. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 444–452. ACM, 2008.

[38] Pedro Larranaga and Jose A Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer Science & Business Media, 2002.

[39] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[40] Marcus A Maloof. Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 workshop on learning from imbalanced data sets II*, volume 2, pages 2–1, 2003.

[41] David J Miller and John Browning. A mixture model and em-based algorithm for class discovery, robust classification, and outlier rejection in mixed labeled/unlabeled data sets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(11):1468–1483, 2003.

[42] Tom M Mitchell. Machine learning. wcb, 1997.

[43] Tood K Moon. The expectation-maximization algorithm. *Signal processing magazine, IEEE*, 13(6):47–60, 1996.

[44] Warwick J Nash. *The Population Biology of Abalone (Haliotis Species) in Tasmania: Blacklip Abalone (H. Rubra) from the North Coast and the Islands of Bass Strait.* Sea Fisheries Division, Marine Research Laboratories-Taroona, Department of Primary Industry and Fisheries, Tasmania, 1994.

[45] Spiros Papadimitriou, Hiroyuki Kitagawa, Philip B Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 315–326. IEEE, 2003.

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

[47] Dan Pelleg and Andrew W Moore. Active learning for anomaly and rare-category detection. In *Advances in Neural Information Processing Systems*, pages 1073–1080, 2004.

[48] Clifton Phua, Vincent Lee, Kate Smith, and Ross Gayler. A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*, 2010.

[49] James Pickands III. Statistical inference using extreme order statistics. *the Annals of Statistics*, pages 119–131, 1975.

[50] Leonid Portnoy, Eleazar Eskin, and Sal Stolfo. Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*. Citeseer, 2001.

[51] Marcel Prastawa, Elizabeth Bullitt, Sean Ho, and Guido Gerig. A brain tumor segmentation framework based on outlier detection. *Medical image analysis*, 8(3):275–283, 2004.

[52] Ronaldo C Prati, Gustavo EAPA Batista, and Maria Carolina Monard. A study with class imbalance and random sampling for a decision tree learning system. In *Artificial Intelligence in Theory and Practice II*, pages 131–140. Springer, 2008.

[53] John Ross Quinlan, Paul J Compton, KA Horn, and Leslie Lazarus. Inductive knowledge acquisition: a case study. In *Proceedings of the Second Australian Conference on Applications of expert systems*, pages 137–156. Addison-Wesley Longman Publishing Co., Inc., 1987.

[54] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

[55] Ida Ruts and Peter J Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics & Data Analysis*, 23(1):153–168, 1996.

[56] Erich Schubert, Alexander Koos, Tobias Emrich, Andreas Züfle, Klaus Arthur Schmid, and Arthur Zimek. A framework for clustering uncertain data. *PVLDB*, 8(12):1976–1987, 2015.

[57] David W Scott. Outlier detection and clustering by partial mixture modeling. In *COMPSTAT 2004—Proceedings in Computational Statistics*, pages 453–464. Springer, 2004.

[58] Jack W Stokes, John C Platt, Joseph Kravis, and Michael Shilman. Aladin: Active learning of anomalies to detect intrusions. *Technique Report. Microsoft Network Security Redmond, WA*, 98052, 2008.

[59] Yuchun Tang, Yan-Qing Zhang, Nitesh V Chawla, and Sven Krasser. Svms modeling for highly imbalanced classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(1):281–288, 2009.

[60] David MJ Tax and Robert PW Duin. Support vector domain description. *Pattern recognition letters*, 20(11):1191–1199, 1999.

[61] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[62] Kai Ming Ting. *Inducing cost-sensitive trees via instance weighting*. Springer, 1998.

[63] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

[64] L. Torgo. *Data Mining with R, learning with case studies.* Chapman and Hall/CRC, 2010.

[65] Mark PJ Van der Loo. Distribution based outlier detection for univariate data. *Statistics Netherlands*, 10003, 2010.

[66] Seth van Hooland, Ruben Verborgh, and Max De Wilde. Cleaning data with OpenRefine. In Adam Crymble, Patrick Burns, and Nora McGregor, editors, *The Programming Historian.* August 2013.

[67] Konstantinos Veropoulos, Colin Campbell, Nello Cristianini, et al. Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on AI*, pages 55–60, 1999.

[68] Zhenxia Xue, Youlin Shang, and Aifen Feng. Semi-supervised outlier detection based on fuzzy rough c-means clustering. *Mathematics and Computers in simulation*, 80(9):1911–1921, 2010.

[69] Cui Zhu, Hiroyuki Kitagawa, Spiros Papadimitriou, and Christos Faloutsos. Obe: outlier by example. In *Advances in Knowledge Discovery and Data Mining*, pages 222–234. Springer, 2004.

[70] Cui Zhu, Hiroyuki Kitagawa, Spiros Papadimitriou, and Christos Faloutsos. Outlier detection by example. *Journal of Intelligent Information Systems*, 36(2):217–247, 2011.

[71] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.