Universiteit
Leiden
The Netherlands

Trustworthy anomaly detection for smart manufacturing

Li, Z.

Citation
Li, Z. (2025, May 1). *Trustworthy anomaly detection for smart manufacturing*. *SIKS Dissertation Series*. Retrieved from https://hdl.handle.net/1887/4239055

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/4239055

Note: To cite this publication please use the final published version (if applicable).

Chapter 4

Explainable Contextual Anomaly Detection using Quantile Regression Forests

Authors: **Zhong Li**, Matthijs van Leeuwen

Published in Data Mining and Knowledge Discovery, 37(6), 2517-2563.

Abstract

Traditional anomaly detection methods aim to identify objects that deviate from most other objects by treating all features equally. In contrast, contextual anomaly detection methods aim to detect objects that deviate from other objects within a *context* of similar objects by dividing the features into contextual features and behavioral features. In this paper, we develop connections between dependency-based traditional anomaly detection methods and contextual anomaly detection methods. Based on resulting insights, we propose a novel approach to inherently interpretable contextual anomaly detection that uses Quantile Regression Forests to model dependencies between features. Extensive experiments on various synthetic and real-world datasets demonstrate that our method outperforms state-of-the-art anomaly detection methods in identifying contextual anomalies in terms of accuracy and interpretability.

4.1 Introduction

According to the well-known definition of [127], an anomaly is an object that is notably different from most remaining objects. [31] subdivided anomalies into three types: point anomalies (an object is considered anomalous when compared against the rest of objects), contextual anomalies (an object is anomalous in a specific context), and collective anomalies (a collection of objects is anomalous with respect to the entire dataset). The analysis of anomalies has a wide range of applications, such as in network security [128], bioinformatics [129], fraud detection [130], and fault detection and isolation [131].

Anomaly analysis consists of two equally important tasks: anomaly detection and anomaly explanation. A wealth of 'shallow' machine learning based methods, i.e., not based on deep learning, have been proposed to detect anomalies [31]. More recently, many deep learning based anomaly detection methods have also been developed [36]. However, deep learning based anomaly detection methods are notoriously known as not being interpretable, in the sense that generally both the model itself is non-transparent and the resulting anomaly scores are challenging to interpret without the use of a post-hoc explainer. In this paper it is especially the latter that we consider to be problematic, as post-hoc explanations often rely not only on the model but also on the specific explainer used. In addition, deep learning methods typically require large amounts of data and training the models is a time-consuming process. In many real-world applications, however, 'native' interpretability (i.e., without posthoc explainer) may be required, and limited data and/or computation time may be available. For these reasons, in this paper we restrict our focus to 'shallow' machine learning based methods. In correspondence with this choice, we focus on settings where the amount of data is smaller than is typically required to learn accurate deep models. Most existing shallow methods only consider point anomaly detection, largely ignoring contextual anomaly detection. Moreover, anomaly explanation has received very limited attention. In this paper, we address both the problem of contextual anomaly detection and that of anomaly explanation, for small to moderately sized tabular data having categorical and/or quantitative features.

Shallow anomaly detectors are typically categorized into distance-based, density-based, and distribution-based approaches [132]. Distance-based and density-based methods use knowledge about the spatial proximity of objects to identify anomalies, while distribution-based methods use knowledge about the distribution of the data to detect anomalies. These methods work under the assumption that *objects having a*

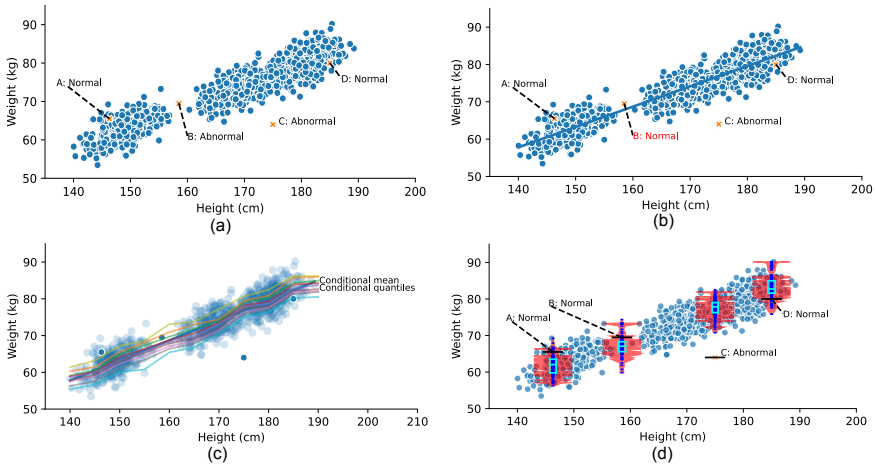

Figure 4.1: Different types of anomaly detection. (a) Distance- and density-based methods both consider object B to be abnormal, because it is far away from other objects and in a low-density region. (b) Dependency-based methods model the relationship between height and weight, and consequently consider object B to be normal. (c) Conditional quantiles provide more information about the conditional distribution than just the mean, and (d) can be used to visualize—using beanplots (or, rather, a variation of a beanplot combined with boxplots, see Chapter 4.5.2 for details)—why a certain object is considered (ab)normal.

large distance or different density from their spatial neighbors are anomalous or *objects that take rare values under a marginal or joint distribution of features are anomalous*, respectively. Using either of these assumptions may lead to false positives though. For example, as shown in Figure 4.1(a), these methods will mistakenly identify object B as an anomaly if the aim is to detect people that are over- or underweight.

As this is often undesirable, we aim to define and detect anomalies from another perspective, that is, we assume that *objects that violate a dependency, i.e., a relationship between features, are anomalous*. This is the idea behind dependency-based anomaly detection [133], which is not new but has received limited attention compared to other types of anomalies. It leverages the intrinsic structure and properties of the data to find potentially more relevant anomalies. For example, Figure 4.1(b) shows that this approach will identify object B as normal by modeling the dependency between height and weight.

Traditional anomaly detection techniques—including distance-, density-, and dependency-based methods—treat all features equally when identifying anomalies. However, in domains such as healthcare, sensor networks, and environment protection, some features should never be used directly to quantify anomalousness. For instance, forest

fire detection systems should not treat 'deviating' values of latitude, longitude, date, and time as an indication of an anomaly. We should not simply discard these features either though, as they may contain relevant information. For example, the 'normal' temperature may be higher for certain regions than for others. This motivates us to investigate *contextual* anomaly detection, which can take such extra features into account.

Contextual anomaly detection assumes that *an object is anomalous if it strongly deviates from objects within its 'context' of similar objects.* The apparent contradiction in this assumption is explained by the division of the features into two disjoint subsets, i.e., contextual features and behavioral features. The contextual features are only used to define the contexts, using some similarity measure, while the behavioral features are only used to determine whether an object deviates from other objects within its context. Domain knowledge often leads to a natural division between contextual and behavioral features.

We observe that both contextual and dependency-based anomaly detection methods identify anomalies by explicitly or implicitly exploring dependencies between features. Concretely, dependency-based anomaly detection methods model dependency relationships between all features explicitly, while contextual anomaly detection methods model dependency relationships between behavioral and contextual features implicitly or explicitly. As far as we know this connection has not yet been pointed out in the literature.

Approach and contributions. In this paper, we introduce an approach for contextual anomaly detection and explanation that integrates the core principle of dependency-based anomaly detection into contextual anomaly detection to obtain a very accurate approach. As is common, we use regression analysis to model dependencies. Existing methods for dependency-based and contextual anomaly detection that use regression, however, typically only estimate the conditional mean of the response variable and directly interpret that as 'normal' value. This strongly limits how well anomalies can be detected, as the conditional mean provides very limited information about the conditional distribution. We therefore use *quantile regression*, which can model a conditional distribution in much more detail by estimating conditional quantiles. Figure 4.1(c) shows how conditional quantiles provide more information about the relationship between weight and height than a conditional mean could.

More specifically, a subset of the features, dubbed contextual features, are used to define the context of an object, while the remaining features, dubbed behavioral features, are used for detecting deviations within a context. In this paper we assume

that the contextual features can be mixed but all behavioral features are numerical. Given this context and our aims, we use Quantile Regression Forests [134] to perform predictions for each behavioral feature and obtain corresponding uncertainty quantifications. By summing the quantified uncertainties (with a wider quantile interval representing a higher level of uncertainty) for all individual behavioral features, we obtain the anomaly score for a data instance. By attributing parts of the anomaly score to individual behavioral features, the approach intrinsically provides explanations in the sense that it can convey to which extent which features contributed to making the instance an anomaly. This offers advantages to post-hoc explanation methods such as SHAP [135], which we do not consider 'attributable' for the following two reasons. First, the post-hoc explanation may not match the information/rationale used by the model to detect an anomaly [70]. Second, it has recently been shown [136] that attribute-based explanations cannot be both recourse sensitive and robust, which is a good reason to avoid such explainers when possible and use a 'native' attribution-based method instead.

As far as we are aware, we are the first to use quantile regression for dependency-based or contextual anomaly detection. Specifically, we choose to employ Quantile Regression Forests for two reasons. First, it can model both linear and non-linear dependency between features. Second, in the paper that introduced the method it was empirically shown to outperform other conditional quantile estimators in most cases. Figure 4.1(d) shows how the estimated conditional quantiles can be used to approximate the conditional probability density at different locations, which we can use to accurately detect anomalies. Moreover, the quantiles are helpful to explain why an object is considered an anomaly without having to explicitly refer to other objects.

The main contributions of our work can be summarized as follows: (1) We identify a connection between dependency-based traditional anomaly detection methods and contextual anomaly detection methods, and exploit this observation to introduce a novel high-level approach to contextual anomaly detection and explanation; (2) We instantiate this generic approach using quantile regression (and Quantile Regression Forests specifically) for anomaly detection and a beanplot-based visualization for anomaly explanation; and (3) We perform extensive experiments on synthetic and real-world datasets to empirically demonstrate the effectiveness, and interpretability of the proposed method when compared to state-of-the-art methods.

The remainder of this paper is organized as follows. Chapter 4.2 discusses related work, both in contextual and traditional anomaly detection. Chapter 4.3 introduces notation, formalizes the problem, and presents the high-level approach that we pro-

pose. Chapter 4.4 then describes some technical preliminaries, most notably Quantile Regression Forests. Chapter 4.5 introduces QCAD, our proposed method for Quantile-based Contextual Anomaly Detection and Explanation that instantiates the high-level approach. Chapter 4.6 empirically compares QCAD to its competitors, and provides a case study that investigates the use of QCAD to find exceptional football players from data. Chapter 4.7 concludes the paper.

4.2 Related Work

We first discuss related work on contextual anomaly detection and explanation, and then proceed with the two most closely related types of traditional anomaly detection: dependency-based and subspace-based anomaly detection.

4.2.1 Contextual Anomaly Detection and Explanation

Contextual anomaly detection has received particular attention in spatial data [137], temporal data [138], and spatio-temporal data [139], where spatial and/or temporal features are used to define contexts. These methods are not directly applicable to other domains, where the contexts are defined by other types of features; 'generic' contextual anomaly detection has received limited attention in the community.

CAD [140] is a seminal work that introduced generic contextual anomaly detection. It assumes a user-specified partition of features into contextual (called 'environmental') and behavioral (called 'indicator') features, and uses Gaussian Mixture Models to fit the distributions of the contextual and behavioral feature spaces. Dependencies between contextual features and behavioral features are then learned by means of 'mapping functions', and an object is considered anomalous if it violates the learned functions. For this to work CAD assumes that both the contextual and behavioral features consist of an unknown number of multiple Gaussian components, which may be a strong assumption in practice. Further, CAD can only handle numerical features and is computationally very expensive. Our proposed method makes no assumptions about the distribution of the features, can deal with mixed contextual features and numerical behavioral features, and we will show empirically that it is computationally more efficient than CAD while achieving a higher detection accuracy.

ROCOD [141] is also closely related, and uses local and global models of expected behavior to describe the dependencies between contextual and behavioral features. Concretely, standard regression models such as CART are used to learn global pat-

terns, with contextual features as predictor variables and behavioral features as response variables. Local patterns are computed based on the means of behavioral feature values of an object's neighbors. An object's actual value is compared to the local and global pattern, and the weighted average of these differences forms the anomaly score. As the conditional mean describes only one aspect of a conditional distribution, and is not necessarily the point with the highest probability of occurrence (i.e., the mode). To address this, our method employs quantile regression analysis to estimate conditional quantiles, which provide a much more complete description of the conditional distribution. As a result, our method empirically outperforms ROCOD in terms of accuracy.

With the increasing use of anomaly detection algorithms in safety-critical domains, such as healthcare and manufacturing, the ethical and regulatory obligations to provide explanations for the high-stakes decisions made by these algorithms has become more pronounced [70]. Existing contextual anomaly detection do not provide such explanations though, and non-trivial modifications would be needed for them to do so. Specifically, CAD [140] computes the anomaly score of a data instance by measuring its deviation from the mapping functions that are learned from the majority of data instances. This approach poses a challenge in generating intrinsic explanations, as it does not allow for the attribution of the anomaly score to individual features. Further, ROCOD [141], the other known method for contextual anomaly detection, calculates the weighted average of an instance' s differences to the learned local and global patterns as its anomaly score. The local pattern is obtained using its neighbors, while the global pattern is learned using all instances, making it difficult to associate the anomaly score with individual features.

Despite the existence of numerous methods for explaining anomalies, such as those outlined in recent (survey) papers [142, 70, 143], there has been very limited research on contextual anomaly explanation. Particularly, COIN [144] explains outliers by reporting their outlierness score, the features that contribute to its abnormality, and a contextual description of its neighborhoods. COIN treats all features equally though, while our method divides features into contextual and behavioral features. Further, we develop a visualization that helps explain contextual anomalies in addition to reporting the overall anomaly score, feature importance, contextual neighbors, and individual anomaly score for each behavioral feature.

The following methods are less relevant because they consider slightly different problems. [145] construct a non-parametric graph-based method for conditional anomaly detection, which only addresses the problem of a single categorical behavioral fea-

ture. [146] first identify anomalies on behavioral features, and then refine the detected anomalies by clustering all objects on contextual features. [147] detect group anomalies from multidimensional categorical data. [148] present a contextual anomaly detection framework dedicated to categorical behavioral features which also considers the dependencies between behavioral features. Moreover, [149] apply robust metric learning on contextual features to find more meaningful contextual neighbors and then leverage k-NN kernel regression to predict the behavioral feature values. [150] develop ConOut to automatically find and incorporate multiple contexts to identify and interpret outliers.

4.2.2 Traditional Anomaly Detection

Although traditional anomaly detection considers a problem that is different from contextual anomaly detection, dependency-based and subspace-based anomaly detection leverage techniques that are related to our method.

Dependency-based Anomaly Detection

Dependency-based anomaly detection aims to identify anomalies by exploring dependencies between features. [151] explores the dependency between non-target and target features to identify and correct possible noisy data points. To detect networking intrusions, [152] present Cross-Feature Analysis to capture the dependency patterns between features in normal networking traffic. To detect disease outbreaks, [153] propose to explore the dependency between features using a Bayesian network. [154] propose to detect anomalies by using an ensemble of models, with each model exploring the dependency between a response feature and other features. [155] use Linear Gaussian Bayesian networks to detect anomalies that violate causal relationships.

LoPAD [156] first uses a Bayesian network to find the Markov Blankets of each feature. Then, a predictive model (e.g., CART) is learned for each individual feature as response variable, with its Markov Blankets as predictor variables. Given an object, LoPAD computes the Euclidean distance between its actual value and predicted value (i.e., conditional mean) as its anomaly score, for each feature. The resulting anomaly scores are normalized and summed to obtain the final anomaly score for an object. The method does not distinguish contextual and behavioral features and—like ROCOD—uses conditional means to represent the conditional distribution. Consequently, LoPAD cannot (accurately) detect contextual anomalies.

Subspace-based Anomaly Detection

Subspace-based anomaly detection seeks to find anomalies in part of the feature space. Specifically, [157] propose SOD to identify outliers in varying subspaces. Concretely, they construct axis-parallel subspaces spanned by the neighbors of a given object. On this basis, they investigate whether this object deviates significantly from its neighbors on any of these subspaces. Furthermore, [158] extend this work to determine whether an object is anomalous on arbitrarily oriented subspaces spanned by its neighbors. [159] propose to find subspaces with strong mutual correlations and then identify anomalies on these subspaces. Finally, [160] use archetype analysis to project the feature space into various subspaces with linear correlations based on nearest neighbors. On this basis, they explore outliers by ensembling the results obtained on relevant subspaces. Overall, these methods pursue to identify anomalies in a subset of features, but treat all features equally and are thus not suitable to identify contextual anomalies.

4.3 Contextual Anomaly Detection and Explanation

We first introduce the necessary terminology and notations, and illustrate this with the running example depicted in Table 4.1. A dataset $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_N\}$ contains N instances (or data points) over the set of features (a.k.a. attributes or variables) denoted by $\mathbf{F} = \{\mathbf{f}^1, \ldots, \mathbf{f}^j, \ldots, \mathbf{f}^D\}$. x_i^j denotes the value of the i-th object, $\mathbf{x}_i$, for the j-th feature, $\mathbf{f}^j$. In the running example, we have $N = 16$, $D = 6$, $\mathbf{F} = \{Latitude, Longitude, Season, Temperature, Rain, Wind\}$, and $x_1^{Rain} = 69$.

We assume that feature set $\mathbf{F}$ is divided into two disjoint feature sets (typically using domain knowledge): a contextual feature set $\mathbf{C} = \{\mathbf{c}^1, ..., \mathbf{c}^p, \ldots, \mathbf{c}^P\}$ and a behavioral feature set $\mathbf{B} = \{\mathbf{b}^1, \ldots, \mathbf{b}^q, \ldots, \mathbf{b}^Q\}$, such that $D = P + Q$, $\mathbf{F} = \mathbf{C} \cup \mathbf{B}$, and $\mathbf{C} \cap \mathbf{B} = \emptyset$. Without loss of generality, we can rearrange the features of an object $\mathbf{x}_i = (x_i^1, ..., x_i^j, ..., x_i^D)$ and represent it as $\mathbf{x}_i = (x_i^1, ..., x_i^P, x_i^{P+1}, ..., x_i^{P+Q}) = (c_i^1, ..., c_i^p, ..., c_i^P, b_i^1, ..., b_i^q, ..., b_i^Q) = (\mathbf{c}_i, \mathbf{b}_i)$, so that $\mathbf{c}_i$ and $\mathbf{b}_i$ denote its contextual and behavioral feature values, respectively. Accordingly, we refer to the space spanned by $\mathbf{C}$ as *contextual space*, i.e., $\mathcal{C} = \mathbf{c}^1 \times ... \times \mathbf{c}^p \times ... \times \mathbf{c}^P$, and to the space spanned by $\mathbf{B}$ as *behavioral space*, i.e., $\mathcal{B} = \mathbf{b}^1 \times ... \times \mathbf{b}^q \times ... \times \mathbf{b}^Q$.

Finally, let Pow denote the powerset, i.e., $\text{Pow}(\mathbf{X}) = \{X \subseteq \mathbf{X}\}$.

Table 4.1: Running example: fictional climate data for Dutch cities. Temperature is measured in degrees Celsius, rain in mm, and wind in miles per hour. The city is not used for anomaly detection. *Latitude*, *Longitude* and *Season* are treated as contextual features, while **Temperature**, **Rain** and **Wind** are considered the behavioral features.

City	*Latitude*	*Longitude*	*Season*	**Temperature**	**Rain**	**Wind**
Leiden	52.16	4.49	Winter	3.0	69	16
Amsterdam	52.37	4.89	Winter	2.9	55	17
Rotterdam	51.92	4.46	Winter	2.7	60	15
Oss	51.45	5.31	Winter	1.1	58	25
Eindhoven	51.44	5.46	Autumn	16.6	62	25
Delft	52.00	4.21	Autumn	16.1	45	19
Utrecht	52.09	5.10	Autumn	18.3	42	20
The Hague	52.07	4.28	Autumn	18.5	49	18
Tilburg	51.33	5.52	Summer	22.1	39	22
Middelburg	51.49	3.61	Summer	20.3	41	23
Arnhem	51.98	5.89	Summer	19.6	48	17
Venlo	51.37	6.17	Summer	21.8	43	35
Emmen	52.46	6.55	Spring	8.3	80	10
Meppel	52.69	6.19	Spring	7.1	27	13
Groningen	53.13	6.34	Spring	4.2	17	19
Leeuwarden	53.10	5.80	Spring	7.2	17	19

4.3.1 Problem Statement

In contextual anomaly detection, contextual features are used to determine the so-called *context* of an object. An object's context is used to estimate whether it is anomalous. The latter is achieved by comparing the object's values for the behavioral features to what is 'normal' within the object's context—if the object's behavioral values strongly deviate, it is flagged as an anomaly.

For contextual anomaly detection to be meaningful, we must assume that *there exist dependencies between the contextual and behavioral data.* If such a relationship does not exist, there is no need to use contextual anomaly detection; one could simply remove the contextual features and reduce the problem to a traditional anomaly detection problem.

We illustrate this using the running example in Table 4.1. We can detect anomalous weather taking into account different regions and seasons by specifying contextual feature set $\mathbf{C} = \{Latitude, Longitude, Season\}$ and behavioral feature set $\mathbf{B} = \{Temperature, Rain, Wind\}$. Each city is now only compared to cities with a similar latitude, longitude, and season, i.e., its context. If a city has values for temperature, rain, and/or wind that strongly deviate from those of the cities in its context, it is marked as anomalous.

For example, Amsterdam and Rotterdam could form the context of Leiden; they are both nearby and we have measurements for the same season. Temperature and wind are similar for all three cities, but there was substantially more rain in Leiden than in Amsterdam and Rotterdam. Hence, for that reason Leiden could be flagged as an anomaly.

To formalize the problem, we introduce a generic 'context function' that maps each possible data point to a subset of the dataset, i.e., its context, based on the data point's contextual features.

Problem 1: Contextual Anomaly Detection Given a dataset $\mathbf{X}$ with a feature set $\mathbf{F} = (\mathbf{C}, \mathbf{B})$, a context function $Context : \mathcal{C} \rightarrow \text{Pow}(\mathbf{X})$, an anomaly detector $Anomaly : \mathcal{B} \times \text{Pow}(\mathbf{X}) \rightarrow \mathbb{R}^+$, and a threshold ϕ, find all data points for which the anomaly scores exceed ϕ and are thereby flagged as anomalous—based on the behavioral features—within their individual contexts, i.e., $\{(\mathbf{c}, \mathbf{b}) \in \mathbf{X} \mid Anomaly(\mathbf{b}, Context(\mathbf{c})) \geq \phi\}$.

Note that the context is determined based only on contextual feature values, and that the anomaly detector may only use the behavioral feature values of the data points in the given context when establishing if the given data point is anomalous or

not.

In practice, analysts are not only interested in identifying anomalies, but also need to know the underlying reasons for why a specific object is reported as anomaly. This leads to the second problem that we consider.

Problem 2: Contextual Anomaly Explanation Given an anomalous object $\mathbf{x} \in \mathbf{X}$ and the context function *Context* and anomaly detector *Anomaly* that were used to detect it, find the behavioral features $\mathbf{B}' \subseteq \mathbf{B}$ for which $\mathbf{x} = (\mathbf{c}, \mathbf{b})$ substantially deviates from $Context(\mathbf{c})$.

4.3.2 Overall Approach

The problem statement in the Chapter 4.3.1 suggests a three-pronged approach based on 1) context generation, 2) anomaly detection, and 3) anomaly explanation. In this subsection we explain and illustrate the overall approach that we propose, using the running example from Table 4.1.

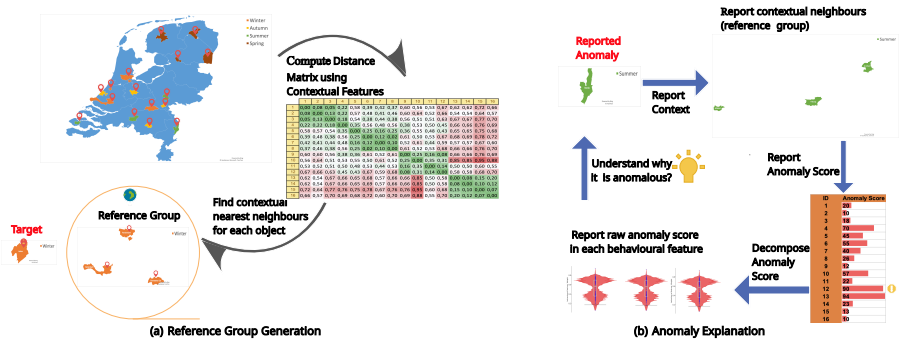

Figure 4.2: Reference Group Generation and Anomaly Explanation. The Reference Group Generation phase computes the distance matrix using only the contextual features, and finds a reference group for each object on this basis. The Anomaly Explanation phase produces an explanation for each identified anomaly by reporting its contextual neighbors, final anomaly score, and raw anomaly scores in each behavioral feature.

Phase 1: Reference Group Generation Many choices are possible for the context function; in this manuscript we choose to use an object's k nearest neighbors, which we refer to as *reference group*. The most important reasons for this choice are that 1) once a global contextual distance matrix has been computed the nearest neighbors of any object can be found relatively quickly; 2) this approach only requires a distance metric to be chosen, which can be defined for any type of data and be adapted to the

problem at hand; and 3) it is generic enough to allow for different uses of the resulting contexts.

Given a dataset $\mathbf{X}$ with contextual feature set $\mathbf{C}$, we first compute the distance matrix $\mathbf{M}$ between all objects using only the contextual features. Second, for any object $\mathbf{x} \in \mathbf{X}$, we find its k nearest neighbors in *contextual space* based on $\mathbf{M}$. As a result, the k nearest neighbors of $\mathbf{x}$ form a reference group, denoted as $R(\mathbf{x}, k)$, which serves as context in Problems 1 and 2.

For instance, as shown in Figure 4.2 (a), given $\{Latitude, Longitude, Season\}$ as the contextual feature set, we first calculate the distance matrix in the contextual space $Latitude \times Longitude \times Season$. Second, based on the distance matrix, we can find the three nearest neighbors of any object as its reference group[1]. Concretely, the three nearest neighbors for $\mathbf{x}_1$ are $\{\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$, which forms a reference group for $\mathbf{x}_1$, denoted as $R(\mathbf{x}_1, 3)$.

Phase 2: Anomaly Detection Given an object $\mathbf{x} \in \mathbf{X}$ with its reference group $R(\mathbf{x}, k)$, we apply an anomaly detector *Anomaly* to obtain an anomaly score *based only on the behavioral attribute values*. We repeat the above process for all objects, leading to N anomaly scores $S = \{s_1, s_2, ..., s_N\}$. We sort S and use threshold ϕ to obtain a ranked list of contextual anomalies $A = \{\mathbf{a}_1, ..., \mathbf{a}_m, ..., \mathbf{a}_M\}$, with $M \ll N$.

For example, we can apply an anomaly detector on the *behavioral space* $Temperature \times Rain \times Wind$ of $\mathbf{x}_1$ and its reference group $R(\mathbf{x}_1, 3) = \{\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$. As a result, we obtain anomaly score s_1 for $\mathbf{x}_1$. Accordingly, repeating this process leads to a set of anomaly scores $S = \{s_1, ..., s_{16}\}$. In our running example we find $\{\mathbf{x}_4, \mathbf{x}_{12}, \mathbf{x}_{13}\}$ as contextual anomalies, as they behave differently in the behavioral space $Temperature \times Rain \times Wind$ when compared to their corresponding neighbors defined in the contextual space $Latitude \times Longitude \times Season$. For example, Oss ($\mathbf{x}_4$) has relatively stronger winds in winter when compared to its nearest cities Leiden, Amsterdam and Rotterdam.

We require anomaly detectors to be *attributable*, meaning that an anomaly score s generated by an anomaly detector must be decomposable into individual contributions towards anomalousness for each behavioral feature using the anomaly detector's native structure.

[1] We use $k = 3$ for illustrative purposes; in practice, we would have a dataset with far more than 16 records and k would also be much larger.

Phase 3: Anomaly Explanation For each anomalous object $\mathbf{a}_m$, we first report its reference group $R(\mathbf{a}_m, k)$ using the distance matrix obtained in Phase 1. Second, we report its anomaly score s_m, as obtained in Phase 2. Moreover, we decompose s_m into individual contributions from each behavioral feature $\mathbf{b}^q \in \mathbf{B}$, resulting in a list of raw anomaly scores $\mathbf{s}_m = \{s_{m1}, ..., s_{mq}, ..., s_{mQ}\}$. This is possible because s_m is *attributable*. Next, we report the top-h raw anomaly scores in $\mathbf{s}_m$ with their corresponding behavioral features, where h can be specified by the analyst. The top-h behavioral features and raw scores enable analysts to better understand why a specific object is flagged as a contextual anomaly.

For example, Figure 4.2 (b) reports object $\mathbf{x}_{12}$ as an anomaly. To explain why, we first inspect its reference group $R(\mathbf{x}_{12}, 3) = \{\mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}\}$, which contains the three objects most similar to $\mathbf{x}_{12}$. Second, we report its anomaly score, i.e., 90. Next, we decompose the score and report the behavioral features with the highest deviations, e.g., *Wind*. We can interpret the result as: $\mathbf{x}_{12}$ deviates substantially in *Wind* when compared to $\{\mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}\}$, which are all similar in terms of *Latitude*, *Longitude* and *Season*.

4.4 Preliminaries

We first define the conditional mean and conditional quantiles, which we use to motivate the use of conditional quantiles for anomaly detection. Next, we detail quantile regression forests, a model class that can robustly estimate conditional quantiles. Readers familiar with these concepts can skip this section.

4.4.1 From Conditional Mean to Conditional Quantiles

Given a real-valued variable V and a set of variables $\mathbf{U}$, regression analysis aims to model the distribution of V conditioned on $\mathbf{U}$. Specifically, it takes V as the response variable and $\mathbf{U}$ as the predictor variables to construct a model that can be used to predict V based on $\mathbf{U}$. Standard regression uses training data $\{(\mathbf{U}_1, V_1), ..., (\mathbf{U}_n, V_n)\}$ to learn a model that estimates the conditional mean $E(V \mid \mathbf{U} = \mathbf{u})$ and uses that as prediction for V when $\mathbf{U} = \mathbf{u}$ is given. For example, Least Squares Regression fits a model $\hat{\theta}$ by minimizing the expected squared error loss, namely $\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, E\{(V - \hat{V}(\theta))^2) \mid \mathbf{U} = \mathbf{u}\}$.

The conditional mean, however, is only a single statistic of the conditional distribution and is thereby limited in what it can capture. For example, if the conditional

distribution is a multi-modal distribution, the mean is insufficient to describe it (regardless of whether we also consider the standard deviation).

To allow for more comprehensive descriptions of conditional distributions, [161] proposed to estimate *conditional quantiles*. As usual, quantiles are splitting points that divide the range of the probability distribution into consecutive intervals having equal probability. Given a continuous variable V, its conditional α-quantile given $\mathbf{U} = \mathbf{u}$ is defined by $Q_\alpha(\mathbf{u}) = \inf\{v : F(v \mid \mathbf{U} = \mathbf{u}) \geq \alpha\}$, where $F(v \mid \mathbf{U} = \mathbf{u}) = P(V \leq v \mid \mathbf{U} = \mathbf{u})$ is the cumulative distribution function. Conditional quantiles have the potential to describe the full conditional distribution of response variable V.

Existing regression-based anomaly detection methods typically estimate the conditional mean of V by its expected value $\hat{v}$, and then use the Euclidean distance between its actual value and the expected value, i.e., $dist(v, \hat{v})$, as anomaly score. It is hard to interpret this distance though, as the shape and range of the conditional distribution are unknown. In this paper we address this by estimating conditional quantiles instead. In particular, we will show that we can use conditional quantiles to approximate the probability of observing a certain data point in its context, which is then used for the anomaly score.

4.4.2 Quantile Regression Forests

Tree-based regression approaches—such as CART, M5, and Random Forests—are often used to learn both linear and non-linear dependencies. [134] extended the Random Forest to Quantile Regression Forest, which estimates and predicts conditional quantiles instead of means.

Specifically, a quantile regression forest (QRF) is constructed by building an ensemble of K independent decision trees to estimate the full conditional cumulative distribution function of V given $\mathbf{U} = \mathbf{u}$, based on n independent observations $\{(\mathbf{U}_1, V_1), ..., (\mathbf{U}_i, V_i), ..., (\mathbf{U}_n, V_n)\}$. Each $\mathbf{U}_i$ consists of d dimensions. The estimated full conditional distribution can be written as

$$\hat{F}(v|\mathbf{U} = \mathbf{u}) = \hat{P}(V \leq v|\mathbf{U} = \mathbf{u}) = \hat{E}(\mathbb{I}(V \leq v)|\mathbf{U} = \mathbf{u}) = \sum_{i=1}^{n} \omega_i(\mathbf{u})\mathbb{I}(V_i \leq v), \quad (4.1)$$

where $\omega_i(\mathbf{u})$ denotes the weight assigned to observation $(\mathbf{U}_i, V_i)$.

The decision trees that make up a quantile regression forest are constructed similarly to how a random forest is learned, i.e., for each individual tree $m \leq n$ data points are sampled (with replacement), $d' \ll d$ features are randomly selected, and a

criterion such as information gain is used to recursively split the data and tree. Each leaf node keeps all its observations though. K decision trees, namely $T_1(\theta), ..., T_K(\theta)$, are independently grown to form a forest.

Once the forest has been constructed, for a given $\mathbf{U} = \mathbf{u}$ each decision tree $T_j(\theta)$ is traversed to find the leaf node that $\mathbf{u}$ resides in. A weight $\omega_i(\mathbf{u}, T_j(\theta))$ is then computed for each observation $\mathbf{U}_i$, with $i \in \{1, ..., n\}$: if observation $\mathbf{U}_i$ and $\mathbf{u}$ reside in the same leaf node, then $\omega_i(\mathbf{u}, T_j(\theta))$ is defined as 1 divided by the number of samples residing in the leaf node. Otherwise, $\omega_i(\mathbf{u}, T_j(\theta))$ is 0. Next, it takes the average of $\omega_i(\mathbf{u}, T_j(\theta))$ over all decision trees, i.e.,

$$\omega_i(\mathbf{u}) = \frac{1}{K} \sum_{j=1}^{K} \omega_i(\mathbf{u}, T_j(\theta)), \tag{4.2}$$

which is the weight assigned to an observation $\mathbf{U}_i$. Finally, conditional quantile $Q_\alpha(\mathbf{u}) = \inf\{v : F(v \mid \mathbf{U} = \mathbf{u}) \geq \alpha\}$ can be estimated by $\hat{Q}_\alpha(\mathbf{u}) = \inf\{v : \hat{F}(v \mid \mathbf{U} = \mathbf{u}) \geq \alpha\}$. Under reasonable assumptions, [134] proved quantile regression forests to be consistent, i.e.,

$$|\hat{F}(v \mid \mathbf{U} = \mathbf{u}) - F(v \mid \mathbf{U} = \mathbf{u})| \xrightarrow{p} 0, \text{with } n \longrightarrow \infty \tag{4.3}$$

holds pointwise for every $\mathbf{u}$.

4.5 Quantile-based Contextual Anomaly Detection and Explanation

We present an instance of the generic approach for contextual anomaly detection presented in Chapter 4.3.2 that is based on quantile regression forests.

The main idea of our method is to estimate the deviation of an object ' s behavioral values within a given context using uncertainty quantification around predictions, where the predictions are assumed to capture 'normal' behavior. Then, a higher uncertainty implies a higher deviation within the context and thus a higher degree of anomalousness. To this end several approaches could be explored. For example, one might use multi-target regression models—such as Multivariate Random Forests [162]—on all behavioral features, or single-target regression models—such as Random Forests [163]—on individual behavioral features followed by aggregation and conformal inference [164]. In this paper we instead opt to use Quantile Regression Forests [134],

a single-target regression model, because it is (relatively) simple, offering advantages with regard to interpretability, and directly provides uncertainty quantifications. More concretely, we derive intervals for the behavioral features from the underlying quantile regression forests as inherent uncertainty quantifications around predictions, resulting in statistically sound and interpretable measures of degree of anomaly.

In the first phase, we generate reference groups using a distance matrix computed on the *contextual space* of all data points. Specifically, we use Gower's distance [165] (see Chapter 4.6.1 for more detail), to be able to deal with both quantitative and categorical features, and select the k objects having the smallest distances to an object $\mathbf{x}$ as its reference group $R(\mathbf{x}, k)$.

Next, in the second phase, an anomaly score is computed for each individual data point, based on the values in the *behavioral space* of the data point and its reference group. The algorithm is dubbed QCAD—for Quantile-based Contextual Anomaly Detection—and forms the core of our approach; it is introduced in Chapter 4.5.1. Finally, Chapter 4.5.2 describes how the found anomalies are explained by decomposing the anomaly score in the third phase.

Algorithm 3 Quantile-based Contextual Anomaly Detection (QCAD)

Input: Dataset $\mathbf{X}$; contextual feature set $\mathbf{C}$; behavioral feature set $\mathbf{B}$; number of behavioral features Q; number of nearest neighbors k; number of conditional quantiles to estimate n_q; number of trees n_t; maximum number of features used in a tree n_f; minimum number of samples to split a node n_s.
Output: Anomaly score list $\mathbf{S}$

procedure QCAD($\mathbf{X}, \mathbf{C}, \mathbf{B}, k, n_q, n_t, n_f, n_s$)
  $\mathbf{S} \Leftarrow \{\}$
  for $\mathbf{x} \in \mathbf{X}$ **do**
    $R(\mathbf{x}, k) \Leftarrow GetReferenceGroup(\mathbf{X}, \mathbf{C}, \mathbf{x}, k)$
    for $q \in \{1, 2, ..., Q\}$ **do**
      $QRF \Leftarrow LearnQRF(R(\mathbf{x}, k), \mathbf{C}, \mathbf{b}^q, n_q, n_t, n_f, n_s)$
      $s(\mathbf{x}|\mathbf{b}^q) \Leftarrow AnomalyScore(QRF, \mathbf{C}, \mathbf{b}^q, \mathbf{x})$
    end for
    $s(\mathbf{x}) = \frac{1}{Q} \sum_{q=1}^{Q} s(\mathbf{x}|\mathbf{b}^q)$
    $\mathbf{S}.append((\mathbf{x}, R(\mathbf{x}, k), s(\mathbf{x})))$
  end for
  return S
end procedure

4.5.1 Detecting Anomalies with Quantile Regression Forests

Algorithm 3 outlines the QCAD algorithm, which takes a dataset and a number of hyperparameters as input and outputs a list of all data points together with their reference groups and computed anomaly scores. Specifically, we assume that the contextual features can be mixed but all behavioral features are numerical. We will first describe the overall algorithm, and then go into the specifics of the score computation.

Algorithm After initializing the empty score list (Line 2), the algorithm iterates over all data points in dataset $\mathbf{X}$ (Ln 3–11). For each object $\mathbf{x} = (c^1, ..., c^p, ..., c^P, b^1, ..., b^q, ...b^Q) = (\mathbf{c}, \mathbf{b})$ we first obtain the reference group that was computed in the first phase (Ln 4). We then iterate over all features in behavioral feature set $\mathbf{X}$ in order to compute a partial anomaly score for each behavioral feature (Ln 5–8). These partial anomaly scores are summed to obtain the anomaly score for $\mathbf{x}$ (Ln 9), i.e., we assume the behavioral features to all have equal potential to contribute to the overall anomaly score. After this, the data point, its reference group, and its anomaly score are appended to the anomaly score list (Ln 10). Finally, the anomaly score list is returned as output (Ln 12).

Within the inner `for` loop, we first learn a quantile regression forest using behavioral feature $\mathbf{b}^q$ as response variable and the data point's reference group $R(\mathbf{x}, k)$ as training data (Ln 6). All contextual features $\mathbf{C}$ are used as predictor variables for every constructed QRF. We then use the learned quantile regression forest, the features, and the data point to compute the raw partial anomaly score (Ln 7), which we will motivate and explain in detail next.

QRF-based anomaly score As argued in the Introduction and Chapter 4.4.1, taking the distance between a data point and a conditional mean as basis for a contextual anomaly score may be too limiting: this only works if all 'normal' data points reside close to the mean. Instead, we aim to—conceptually—consider the entire conditional probability density function and use the local density of a given data point as a proxy for anomalousness: the lower the density, the higher the anomaly score. Directly accurately estimating the density function is hard though, especially in areas of low density, which are of particular importance to us.

This is where the quantile regression forests come in: given sufficient training data, they accurately learn the conditional cumulative distribution function, which can be queried in inverse form, i.e., through conditional quantiles. When querying a quantile regression forest, this can be done at different granularities. For example, Figure 4.3(a)

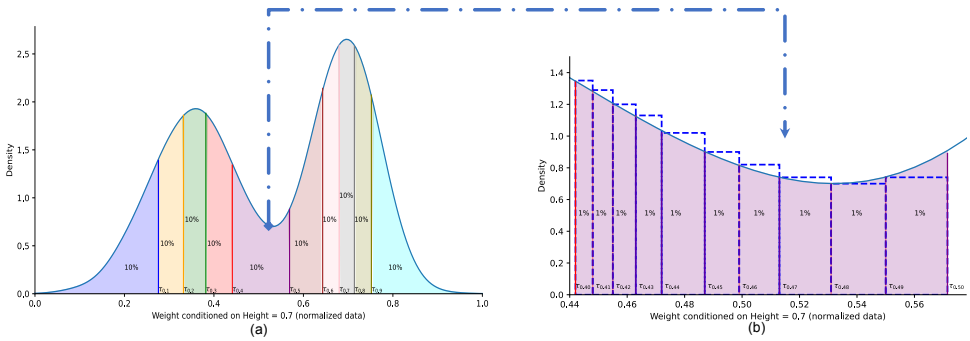

Figure 4.3: (a) Estimated conditional quantiles $\{Q_{0.1}, Q_{0.2}, \ldots, Q_{0.9}\}$ for a behavioral feature conditioned on contextual features. (b) Zooming in on the area between $Q_{0.40}$ and $Q_{0.50}$, we see that percentiles are more likely to be sufficiently detailed than the quantiles in (a). A high-resolution online version improves readability of small fonts.

shows that conditional quantiles $\{Q_{0.1}, Q_{0.2}, \ldots, Q_{0.9}\}$ may very well be insufficient to accurately describe a conditional distribution as they overly smooth the underlying distribution and thus fail to capture the nuances accurately, while Figure 4.3(b) show that conditional *percentiles*, i.e., $\{Q_{0.01}, Q_{0.02}, \ldots, Q_{0.99}\}$, are much more likely to provide sufficient detail.

We use conditional percentiles for our anomaly score, because they provide a high level of granularity while not requiring very large amounts of data to be estimated accurately. As additional benefit, the difference between each two consecutive percentiles is always assessed by a weighted combination of a comparable number of training data points, i.e., at the cost of some smoothing we do not suffer from extremely poor local density estimates in low density areas.

To formally develop our anomaly score, we first define τ_i to be the ith percentile, i.e., $\tau_i = Q_{i/100}, \forall i \in [0, 100]$. For any *percentile interval*, i.e., an interval $[\tau_i, \tau_{i+1}]$ defined by two consecutive percentiles i and $i+1$, we have by definition that it spans exactly 0.01 probability (see Figure 4.3(b)). We could estimate the local density of a percentile interval and use that for our anomaly score, but we aim for a score that becomes larger when a data point is deemed to be more anomalousness.

To this end we define *percentile interval width* w_i as the difference between two consecutive percentiles i and $i+1$, i.e., $\tau_{i+1} - \tau_i$. As such, interval width can be regarded as 'inverse density', meaning that width will increase as the local density decreases. For example, in Figure 4.3(b) we have $\tau_{46} = 0.5$ and $\tau_{47} = 0.513$, which gives $w_{46} = 0.013$. Data points that fall in percentile intervals having relatively large

widths are more likely to be contextual anomalies, as they reside in low-density areas of the conditional distribution.

The basic idea is thus to define the anomaly score for an object $\mathbf{x}$ and behavioral feature $\mathbf{b}^q$ as $w(\mathbf{x}|\mathbf{b}^q)$, i.e., the width of the (QRF-predicted) percentile interval in which the behavioral value of the data point falls. We need to consider a special case though: the actual behavioral feature value b^q may be less than the smallest estimated conditional quantile, i.e., τ_0^q, or greater than the largest estimated conditional quantile, i.e., τ_{100}^q. That is, the actual value may not fall in any estimated conditional percentile interval. To address this we extrapolate beyond τ_0^q and τ_{100}^q, leading to intermediate anomaly score

$$is(\mathbf{x}|\mathbf{b}^q) = \begin{cases} \left(1 + \dfrac{\tau_0^q - b^q}{\tau_{75}^q - \tau_{25}^q}\right) \max(w(\mathbf{x}|\mathbf{b}^q)), \text{ if } b^q < \tau_0^q; \\ \left(1 + \dfrac{b^q - \tau_{100}^q}{\tau_{75}^q - \tau_{25}^q}\right) \max(w(\mathbf{x}|\mathbf{b}^q)), \text{ if } b^q > \tau_{100}^q, \\ w(\mathbf{x}|\mathbf{b}^q), \text{ otherwise,} \end{cases} \tag{4.4}$$

where $\max(w(\mathbf{x}|\mathbf{b}^q))$ represents the maximum interval width of all conditional percentile intervals for the behavioral feature $\mathbf{b}^q$.

Unfortunately, directly using Equation (4.4) as a partial anomaly score would make our approach prone to the so-called **dictator effect**: summing such partial scores for a data point that *strongly* deviates in only a *few* behavioral features would lead to a larger anomaly score than anomalous data points deviating moderately in *many* behavioral features. As a result, data points with few, strong deviations would 'dictate' highest scores; this is undesirable.

To avoid the dictator effect, we truncate the partial anomaly scores to a predefined maximum and arrive at the final partial anomaly score as

$$s(\mathbf{x}|\mathbf{b}^q) = \begin{cases} \dfrac{\eta}{100}, \text{ if } is(\mathbf{x}|\mathbf{b}^q) > \dfrac{\eta}{100}; \\ is(\mathbf{x}|\mathbf{b}^q), \text{ otherwise,} \end{cases} \tag{4.5}$$

where η is a hyperparameter. The rationale for η is as follows: if the conditional distribution is uniformly distributed and the behavioral feature has range $[0, 1]$, then the expected width of any percentile interval width is 0.01 and η can be interpreted as the maximum number of *expected percentile interval widths*. In the experiments we use $\eta = 10$ because of its strong empirical performance (also shown in the ablation study in Appendix 4.7).

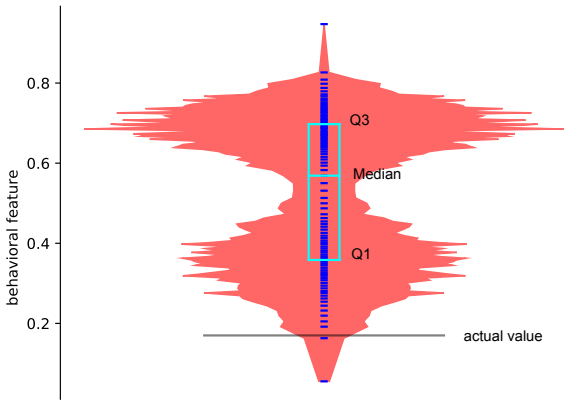

Figure 4.4: Anomaly beanplot giving insight in what the quantile regression forest learned for a particular data point and behavioral feature, and why the data point is (not) considered an anomaly. The short blue lines indicate the conditional percentiles $\tau_0, \tau_1, \ldots, \tau_{100}$ as learned by the QRF (top to bottom), for the given data point and behavioral feature. As in a box plot, the (cyan) box indicates first quartile (τ_{25}), median (τ_{50}), and third quartile (τ_{75}). The wider red area represents probability densities as estimated based on the conditional percentiles. Finally, the black line indicates the actual value that the data point has.

By scaling the behavioral features to $[0, 1]$ before anomaly detection, e.g., using min-max normalization, the estimated conditional percentiles should also lie in this interval and the interval widths obtained for the individual behavioral features should thus be comparable. In turn, this implies they can be summed to obtain the final anomaly score for a data point (Algorithm 3, Line 9).

4.5.2 Explaining Anomalies with Anomaly Beanplots

In the third phase, we provide explanations for the reported anomalies. From the definition of our anomaly score it is clear that it is *attributable*: the overall score can be decomposed into partial scores for individual behavioral features.

For each identified anomaly $\mathbf{a}$, we report its contextual neighbors $R(\mathbf{a})$ and the final anomaly score as computed on Line 9 of Algorithm 3. Further, we report the partial anomaly scores corresponding to the behavioral features, i.e., we report $s(\mathbf{x}|\mathbf{b}^q)$ for all q, ranked from highest to lowest to indicate in which behavioral features the anomaly deviates most.

Since visualization often helps to quickly provide valuable insight, we propose the *anomaly beanplot*, a variant of the beanplot by [166]. Figure 4.4 shows an example, visually depicting the learned conditional percentiles, their interval widths, and the probability densities that can be estimated from those, all for a particular data point and behavioral feature. (Remember that a quantile regression forest is learned on the reference group of a data point, hence the anomaly beanplot for each data point may be different.)

By including the actual value of the data point in the beanplot (as a horizontal black line), the analyst can easily see how its behavioral feature value is positioned relative to those of its reference group, and why and to what extent it contributes to the its anomalousness.

4.6 Experiments

To demonstrate the effectiveness of our overall approach and proposed method QCAD, we conduct experiments on a wide range of synthetic and real-world datasets. We will first explain the choices regarding datasets, baseline algorithms, and evaluation criteria, after which we present both quantitative results and a case study that investigates interpretability and practical utility.

In addition, we demonstrate the robustness of QCAD with regard to the 'number of nearest neighbors' hyperparameter by means of a sensitivity analysis, and conduct several ablation studies to investigate the impact of the hyperparameters. For reasons of space and brevity, the sensitivity analysis and ablation studies are given in Appendices 4.7 and 4.7.

4.6.1 Data

It is challenging to evaluate unsupervised anomaly detection algorithms due to the lack of commonly agreed-upon benchmark data, and down-sampling classification datasets has been criticized for its variation in the nature of the resulting outliers [167, 168].

When evaluating unsupervised *contextual* anomaly detection algorithms, this problem is further compounded by the requirement to have both contextual and behavioral features, and—more importantly—treating those differently [141]. Consequently, a generally accepted approach is to artificially inject contextual anomalies into existing datasets using a perturbation scheme.

Data Preprocessing

To make the datasets suitable to all anomaly detection methods, we need to preprocess them before injecting contextual anomalies. First, we leverage Label Encoding [169] to transform categorical contextual features to numerical form. Second, we employ Min-Max normalization to scale all behavioral features to $[0, 1]$. Min-Max normalization is routinely used in many anomaly detection and generally improves performance [170].

Gower's Distance To be able to calculate the similarity between two data points containing both categorical and numerical features, we can utilize Gower's distance [165]. Specifically, the Gower's distance between data points $\mathbf{c}_i = (c_i^1, ..., c_i^p, ...c_i^P)$ and $\mathbf{c}_j = (c_j^1, ..., c_j^p, ...c_j^P)$ is defined as $1 - \frac{1}{P}\sum_{p=1}^{P} ps_{ij}^p$, where ps_{ij}^p represents the partial similarity between data instances $\mathbf{c}_i$ and $\mathbf{c}_j$ in the p-th dimension. For a numerical feature, $ps_{ij}^p = 1 - \frac{|c_i^p - c_j^p|}{\max(\mathbf{c}^p) - \min(\mathbf{c}^p)}$, with $\max(\mathbf{c}^p)$ and $\min(\mathbf{c}^p)$ denoting the maximum and minimum value of all data points for the p-th feature, respectively. For a categorical feature, $ps_{ij}^p = \mathbb{I}(c_i^p - c_j^p)$, where $\mathbb{I}(\cdot)$ represents the indicator function. Consequently, Gower's distance between two data points is always in $[0, 1]$, with a lower value indicating a larger similarity.

Perturbation Scheme for Outlier Injection

[140] proposed a perturbation scheme to inject contextual anomalies in datasets without ground-truth anomalies, which has become a de-facto standard for the evaluation of contextual anomaly detection [140, 141, 149, 171].

This perturbation scheme, however, has also been criticized for two reasons [140, 172]. First, the objects obtained by simply swapping the feature values are still likely to be contextually normal. Second, some very common types of anomalies cannot be yielded through this perturbation scheme. For example, one cannot obtain extreme values by swapping features values in a clean dataset, whereas most anomaly detection methods assume the training dataset is uncontaminated. To avoid these problems, [172] introduced another perturbation scheme to inject anomalies. We develop a new perturbation scheme by refining this scheme, as follows.

Given a dataset $\mathbf{X}$ containing N data instances with contextual feature set $\mathbf{C} = \{\mathbf{c}^1, ..., \mathbf{c}^P\}$ and behavioral feature set $\mathbf{B} = \{\mathbf{b}^1, ..., \mathbf{b}^Q\}$, we first use Min-Max normalization to scale the behavioral features of all objects to $[0, 1]$ (keeping the contextual features intact), resulting in a new dataset $\tilde{\mathbf{X}}$. Second, to inject m anomalies into $\tilde{\mathbf{X}}$, with $0 < m \ll N$, we select m objects $\mathbf{x}_1, \ldots, \mathbf{x}_m$ uniformly at random from $\tilde{\mathbf{X}}$. For each $\mathbf{x} = (\mathbf{c}, \mathbf{b})$ in $\tilde{\mathbf{X}}$, $\mathbf{c}$ represents the contextual feature values and $\mathbf{b}$

denotes the behavioral feature values. Third, for a selected object $\mathbf{x}_i = (\mathbf{c}_i, \mathbf{b}_i) = (c_i^1, ..., c_i^p, ..., c_i^P, b_i^1, ..., b_i^q, ..., b_i^Q)$ and behavioral feature $\mathbf{b}^q$, we sample a number uniformly at random from $[-0.5, -0.1] \bigcup [0.1, 0.5]$, and then add this random number to the behavioral feature value of $\mathbf{x}_i$, namely b_i^q, resulting in $\hat{b}_i^q$. In the same manner, we repeat this process for each behavioral feature, resulting in $(\hat{b}_i^1, ..., \hat{b}_i^Q)$, or $\hat{\mathbf{b}}_i$. Accordingly, we generate a new object $\tilde{\mathbf{x}} = (\mathbf{c}, \hat{\mathbf{b}})$ as contextual anomaly. Fourth, we repeat the third step for each selected object, leading to m perturbed objects. Fifth and final, we replace the selected objects in the original dataset with their corresponding perturbed objects. To allow extreme values to be injected, we deliberately do not truncate the values outside $[0, 1]$ after adding a random number in each behavioral feature.

Our perturbation scheme has the following advantages. When compared to the swapping perturbation scheme proposed by [140], the objects obtained by our perturbation scheme are very unlikely to remain contextually normal. In addition, our perturbation scheme can—but does not always—lead to extreme values. Note that we do not strictly follow the perturbation scheme proposed by [172] because their method only adds a non-negative number to the behavioral features. On the one hand, sometimes this non-negative number is zero, leading to injecting a normal object as 'anomaly'. On the other hand, sometimes this non-negative number is huge, which makes the injected object (too) easy to detect. Our perturbation scheme avoids these problems by firstly normalizing the behavioral feature values, and then setting more reasonable lower and upper bounds for the perturbation.

Datasets

To evaluate and compare our method on a diverse range of datasets, we generate 10 synthetic datasets and select 20 real-world datasets; their properties are summarized in Tables 4.2 and 4.3 respectively.

We first discuss the synthetic data. To be able to produce data with various forms and degrees of dependencies between behavioral and contextual features, we propose the following generation schemes. For $q \in \{1, ..., Q\}$, we have

(S1) $\mathbf{b}^q = \sum_{p=1}^{Q} (\alpha_{qp} \cdot \mathbf{c}^p) + \boldsymbol{\epsilon}$;

(S2) $\mathbf{b}^q = \sum_{p=1}^{Q} \left(\alpha_{qp} \cdot (\mathbf{c}^p)^3\right) + \boldsymbol{\epsilon}$,

(S3) $\mathbf{b}^q = \sum_{p=1}^{Q} (\alpha_{qp} \cdot \sin(\mathbf{c}^p)) + \boldsymbol{\epsilon}$,

(S4) $\mathbf{b}^q = \sum_{p=1}^{Q} (\alpha_{qp} \cdot \log(\mathbf{1} + |\mathbf{c}^p|)) + \boldsymbol{\epsilon}$,

Table 4.2: Summary of synthetic datasets. $\#Num, \#Cat, \#\mathbf{C}$ and $\#\mathbf{B}$ represent the number of numerical features, the number of categorical/nominal features, the number of contextual features, and the number of behavioral features, respectively. All behavioral features are numerical, the contextual features can be mixed.

Dataset	Scheme	#Num	#Cat	#**C**	#**B**
Syn1	S1	8	2	5	5
Syn2	S2	8	2	5	5
Syn3	S3	8	2	5	5
Syn4	S4	8	2	5	5
Syn5	S5	8	2	5	5
Syn6	S1	19	6	20	5
Syn7	S2	19	6	20	5
Syn8	S3	19	6	20	5
Syn9	S4	19	6	20	5
Syn10	S5	19	6	20	5

(S5) $\mathbf{b}^q = \sum_{p=1}^{Q} \left(\alpha_{qp} \cdot \mathbf{c}^p + \beta_{qp} \cdot (\mathbf{c}^p)^3 + \gamma_{qp} \cdot \sin(\mathbf{c}^p) + \delta_{qp} \cdot \log(\mathbf{1} + |\mathbf{c}^p|)\right) + \boldsymbol{\epsilon}$,

where $\alpha_{qp}, \beta_{qp}, \gamma_{qp}, \delta_{qp} \overset{i.i.d.}{\sim} \mathcal{U}(0,1)$ and are replaced by zero with a probability of 1/3. Further, $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_n, ..., \epsilon_N)^T$, with $\epsilon_n \overset{i.i.d.}{\sim} \mathcal{U}(0, 0.05)$.

In addition, for $p \in \{1, ..., P\}$, $\mathbf{c}^p$ is generated from a Gaussian mixture model with five clusters. If it is numerical, each of the Gaussian centroids is sampled uniformly at random from $[0, 1]$ and the diagonal element of the covariance matrix is fixed at 1/4 of the average pairwise distance between the centroids in each behavioral feature. Otherwise, the centroids are sampled uniformly at random from $\{0, 1, ..., 10\}$ with the same covariance setting. Moreover, every generated number is rounded to an integer to ensure that it is categorical. On this basis, we generate a wide collection of synthetic datasets by varying the generation scheme and the number of contextual features and behavioral features. Sample size is always set to 2000, and the rate of injected anomalies is fixed at 2.5%. See Table 4.2 for an overview.

Next, we employ the above-mentioned perturbation scheme to inject contextual anomalies into 20 real-world datasets. We use these datasets because they are representative of the potential application areas of our QCAD framework. That is, they stem from healthcare & life sciences (e.g., Bodyfat, Heart Failure, Indian River Patient, Hepatitis, Parkinson Telemonitoring, Abalone, Fish Weight, QSAR Fish Toxi-

Table 4.3: Summary of real-world datasets. $\#Num, \#Cat, \#\mathbf{C}$ and $\#\mathbf{B}$ represent the number of numerical features, the number of categorical/nominal features, the number of contextual features, and the number of behavioral features, respectively. All behavioral features are numerical, the contextual features can be mixed.

Dataset	#Num	#Cat	#Samples	#Anomalies (Ratio)	#**C**	#**B**
Abalone	8	1	4177	100 (2.4%)	4	5
AirFoil	6	0	1503	70 (4.7%)	5	1
BodyFat	15	0	252	20 (7.9%)	13	2
Boston	12	2	506	40 (7.9%)	13	1
Concrete	9	0	1030	50 (4.8%)	8	1
ElNino	3	8	20000	200 (1%)	6	5
Energy	10	0	768	50 (6.5%)	8	2
FishWeight	6	1	157	15 (9.5%)	6	1
ForestFires	11	2	517	50 (9.7%)	4	9
GasEmission	11	0	7384	100 (1.4%)	8	3
HeartFailure	7	5	299	30 (10%)	6	6
Hepatitis	11	2	615	30 (4.9%)	3	10
LiverPatient	9	2	579	30 (5.2%)	3	8
Maintenance	18	0	11934	100 (0.8%)	15	3
Parkinson	21	1	5875	100 (1.7%)	20	2
PowerPlant	5	0	9568	100 (1%)	4	1
QSRanking	12	1	475	40 (8.4%)	7	6
SynMachine	5	0	557	50 (8.9%)	4	1
Toxicity	7	0	908	50 (5.5%)	6	1
Yacht	7	0	308	30 (9.7%)	6	1

city), social sciences (e.g., Boston House Price, University Ranking), environmental-protection area (e.g., El Nino, Forest Fires), and engineering (e.g., Gas Turbine CO and NOx Emission, Yacht Hydrodynamics, Condition Based Maintenance of Naval Propulsion Plants, Synchronous Machine, Airfoil Self-Noise, Concrete Compressive Strength, Combined Cycle Power Plant). A summary is given in Table 4.3; each dataset is described in more detail in Appendix 4.7.

4.6.2 Baseline Algorithms and Implementations

We empirically compare our method to state-of-the-art algorithms, including traditional anomaly detection methods (distance-based, density-based, dependency-based, etc.) and contextual anomaly detection methods. For a fair comparison, we select the following anomaly detectors, which all return an anomaly score—rather than a binary outcome—for each data instance.

- Local Prediction Approach to Anomaly Detection (LoPAD) by [156], which is the state-of-the-art dependency-based traditional anomaly detector;
- Conditional Outlier Detection (CAD) by [140], which was the first anomaly detector dedicated to identify contextual anomalies;
- Robust Contextual Outlier Detection (ROCOD) by [141], which is the state-of-the-art contextual anomaly detector;
- Isolation Forest (IForest) by [77], which is one of the state-of-the-art isolation-based traditional anomaly detectors;
- Local Outlier Factor (LOF) by [173], which is one of the state-of-the-art density-based traditional anomaly detectors;
- k-NN anomaly detector by [174], which is one of the state-of-the-art distance-based traditional anomaly detectors;
- Anomaly detector using axis-parallel subspaces (SOD) by [157], which is one of the state-of-the-art subspace-based traditional anomaly detectors; and
- Histogram-Based Outlier Score (HBOS) by [85], which is one of the state-of-the-art histogram-based traditional anomaly detectors.

We implemented and ran all algorithms in Python 3.8 on a computer with Apple M1 chip 8-core CPU and 8GB unified memory. For classical algorithms such as IForest,

LOF, k-NN, SOD, and HBOS, we use their publicly available implementations in PyOD [107] with their default settings. Unfortunately, for LoPAD, CAD and ROCOD no implementation was publicly available. For LoPAD, we first use the 'bnlearn' package [175] in R to find the Markov Blankets of each dataset based on Fast-IAMB [176]. Next, we implement the LoPAD algorithm in Python using CART [177] as the prediction model with bagging size 200. For CAD, we implement the CAD-GMM-Full algorithm as recommended by [140], with default settings except for parameter 'number of Gaussian component'; if this parameter would be set to its default 30, it would take more than one month to finish all the experiments on our computer. In addition, preliminary experiments show that the difference in results when this parameter is set to 30 and 5 respectively is negligible for most datasets. We therefore set this parameter to 5 in all experiments. We implemented ROCOD in Python with its default settings. One important parameter, namely the distance threshold used to find neighbors, is not discussed in [141] though. For different datasets and distance metrics, it is hard to obtain a single best value for this parameter and preliminary experiments revealed that ROCOD is sensitive to this parameter. Nevertheless, we also observed that it often achieves relatively good results when the distance threshold used to find neighbors is set to 0.9; hence, we decided to set this parameter to 0.9 by default.

QCAD Parameters Setting

As summarized in Table 4.4, the QCAD algorithm also requires several parameters to be set. First, in our contextual anomaly framework, we need to set the number of nearest neighbors (k) used to generate reference group. Second, when creating quantile regression forests, we need to specify the following parameters: the number of trees, the number of maximal features used to construct a tree, and the number of minimal sample size to split in a node of tree. Third, we need to specify the number of conditional quantiles (l) to estimate for an object in each behavioral feature. Last, we also need to set the number of features (h) used to generate explanations. We set these parameters as follows.

- The number of nearest neighbors (k): the sensitivity analysis (see appendix) indicates that our approach is robust with respect to this parameter as long as its value is not overly small. By default, we set this parameter to $\min(N/2, 500)$, where N is the sample size.
- The number of conditional quantiles to estimate (n_q): theoretically, an increase

Table 4.4: Summary of parameters involved in QCAD. Particularly, **Value** represents the values that we recommend to use in experiments.

Symbol	Meaning	Value
k	number of nearest neighbors	$\min(N/2, 500)$
n_q	number of conditional quantiles to estimate	100
n_t	number of trees used to construct a QRF	100 or 10
n_f	number of maximal features used to construct a tree	$\|\mathbf{C}\|$
n_s	minimum number of samples to split a node	10
h	number of features used to generate explanations	$\min(\|\mathbf{B}\|, 3)$

in this number will result in better performance in terms of accuracy, at the expense of a larger running time. However, preliminary experiments show that increasing this number beyond 100 will only produce slightly better results. Therefore, we set it to 100 by default.

- The number of trees used to construct a quantile regression forest (n_t): in theory, a larger number of trees will produce better performance in terms of accuracy, but at the cost of a larger running time. We empirically found that 100 trees usually gives good results and further increasing the number of trees leads to negligible improvement. Due to time constraints, we set this number to 10 in all experiments in this paper.

- The number of maximal features used to construct a tree in a quantile regression forest (n_f): [134] demonstrated the stability of quantile regression forest on this parameter, and set this parameter to 1/3 of the number of variables in their experiments. However, in our experiments, sometimes the number of contextual features is less than 3. To render quantile regression forests applicable on various datasets, we set this parameter to the number of all variables by default.

- The minimal sample size for the node of a tree to be split (n_s): as indicated in [134], different values of this parameter do not seem to have much effect on the results, and our preliminary experiments are also in line with this statement. Therefore, we set this number to 10 by default, as also used in [134].

- The number of features used to generate explanations (h): This parameter is set to $\min(|\mathbf{B}|, 3)$ by default. However, the end-users can set this parameter according to their preferences as long as its value is between 0 and $|\mathbf{B}|$, where $|\mathbf{B}|$ represents the number of behavioral features.

For reproducibility, we make all code and datasets publicly available[2].

4.6.3 Evaluation Criteria

PRC AUC [141, 172], ROC AUC [178, 179, 180], and Precision@n [181] are widely used for the evaluation and comparison of anomaly detection methods that generate a full list of anomaly scores for all observations. They are defined as follows:

- Receiver Operating Characteristic (ROC), which is obtained by plotting the true positive rate (y-axis) versus the false positive rate (x-axis) at various threshold settings. The area under this curve, namely ROC AUC, is a threshold-agnostic performance measure widely used in anomaly detection;
- Precision-Recall Curve (PRC), which is created by plotting precision (y-axis) against recall (x-axis) at various threshold settings. The area under this curve, namely PRC AUC, is another widely-used, threshold-agnostic performance measure, and is also called Average Precision;
- Precision at n, or P@n, is defined as the precision of the observations ranked among the top-n, where $n \in \{1, 2, ..., N\}$. In our experiments, we set n to the number of injected contextual anomalies.

For completeness: precision is defined as $\frac{\#\{\text{Real anomalies}\}\cap\{\text{Reported anomalies}\}}{\#\{\text{Reported anomalies}\}}$, while recall is defined as $\frac{\#\{\text{Real anomalies}\}\cap\{\text{Reported anomalies}\}}{\#\{\text{Real anomalies}\}}$. We perform ten independent trials of injecting contextual anomalies on each dataset and report the means and standard deviations of each of the three evaluation criteria.

4.6.4 Anomaly Detection Performance

Results on 10 synthetic datasets and 20 real-world datasets are presented in Tables 4.5, 4.6, and 4.7, where the first table concerns synthetic data, and the latter two tables concern real-world data.

From Table 4.5, we observe that QCAD generally dominates other methods in terms of anomaly detection accuracy according to the average ranks. More specifically, QCAD achieved the best results on 9 out of 10 synthetic datasets in terms of PRC AUC and ROC AUC, and on all datasets in terms of Precision@n. On Syn8, QCAD is on par with its best competitor (i.e., CAD) in terms of ROC AUC, whereas

[2]https://github.com/ZhongLIFR/QCAD

Table 4.5: Performance in terms of PRC AUC, ROC AUC, and P@n, on synthetic data, with 10 independent runs of injecting contextual anomalies into each dataset. For each anomaly detector on each dataset, the mean value and standard deviation of each evaluation criterion is presented. The best results obtained on each dataset are highlighted in bold.

		QCAD	LoPAD	ROCOD	CAD	IForest	LOF	k-NN	SOD	HBOS
Syn1	PRC AUC	**1.00**±0.00	0.03±0.00	0.88±0.03	0.94±0.15	0.32±0.08	0.04±0.01	0.05±0.01	0.29±0.03	0.18±0.03
	ROC AUC	**1.00**±0.00	0.49±0.04	0.99±0.00	0.94±0.16	0.95±0.01	0.64±0.05	0.72±0.03	0.97±0.00	0.90±0.02
	P@n	**0.99**±0.01	0.04±0.00	0.79±0.03	0.89±0.31	0.37±0.08	0.05±0.03	0.05±0.02	0.20±0.07	0.22±0.06
Syn2	PRC AUC	**1.00**±0.00	0.03±0.00	0.22±0.02	0.94±0.03	0.15±0.03	0.04±0.01	0.05±0.01	0.29±0.02	0.38±0.08
	ROC AUC	**1.00**±0.00	0.49±0.02	0.95±0.00	**1.00**±0.00	0.93±0.01	0.66±0.03	0.71±0.02	0.97±0.00	0.95±0.01
	P@n	**0.98**±0.02	0.03±0.01	0.13±0.03	0.94±0.02	0.09±0.05	0.04±0.02	0.04±0.02	0.27±0.06	0.36±0.08
Syn3	PRC AUC	**1.00**±0.00	0.02±0.00	0.69±0.04	0.96±0.02	0.31±0.05	0.07±0.01	0.06±0.01	0.34±0.04	0.26±0.05
	ROC AUC	**1.00**±0.00	0.47±0.06	0.97±0.01	0.99±0.01	0.93±0.02	0.77±0.02	0.76±0.02	0.97±0.00	0.92±0.01
	P@n	**0.97**±0.01	0.02±0.02	0.60±0.04	0.94±0.02	0.34±0.06	0.09±0.03	0.06±0.02	0.34±0.06	0.29±0.05
Syn4	PRC AUC	**1.00**±0.00	0.03±0.01	0.90±0.03	0.99±0.01	0.38±0.08	0.05±0.01	0.06±0.01	0.44±0.04	0.32±0.04
	ROC AUC	**1.00**±0.00	0.52±0.02	**1.00**±0.00	**1.00**±0.00	0.95±0.01	0.72±0.03	0.78±0.03	0.99±0.00	0.93±0.01
	P@n	**1.00**±0.00	0.03±0.03	0.82±0.04	0.98±0.02	0.43±0.06	0.05±0.03	0.08±0.03	0.49±0.07	0.37±0.05
Syn5	PRC AUC	**1.00**±0.00	0.02±0.00	0.26±0.03	**1.00**±0.00	0.15±0.02	0.04±0.00	0.05±0.00	0.22±0.01	0.21±0.03
	ROC AUC	**1.00**±0.00	0.52±0.03	0.96±0.00	**1.00**±0.00	0.93±0.01	0.61±0.03	0.70±0.02	0.96±0.00	0.94±0.01
	P@n	0.99±0.01	0.02±0.02	0.21±0.04	**1.00**±0.00	0.11±0.06	0.04±0.02	0.05±0.03	0.12±0.03	0.25±0.05
Syn6	PRC AUC	**0.99**±0.01	0.03±0.01	0.97±0.01	0.91±0.14	0.18±0.04	0.03±0.00	0.03±0.00	0.03±0.00	0.40±0.07
	ROC AUC	**1.00**±0.00	0.49±0.06	**1.00**±0.00	0.94±0.16	0.88±0.02	0.52±0.03	0.51±0.04	0.53±0.05	0.93±0.01
	P@n	**0.95**±0.02	0.03±0.03	0.91±0.03	0.83±0.29	0.26±0.06	0.03±0.03	0.04±0.03	0.02±0.02	0.42±0.06
Syn7	PRC AUC	**1.00**±0.00	0.02±0.00	0.68±0.07	0.98±0.02	0.10±0.03	0.03±0.00	0.03±0.00	0.03±0.01	0.28±0.05
	ROC AUC	**1.00**±0.00	0.48±0.05	0.98±0.01	**1.00**±0.00	0.84±0.03	0.49±0.04	0.50±0.03	0.55±0.03	0.94±0.01
	P@n	**0.99**±0.01	0.02±0.02	0.64±0.05	0.97±0.03	0.12±0.05	0.02±0.02	0.01±0.02	0.02±0.03	0.32±0.04
Syn8	PRC AUC	0.77±0.04	0.02±0.00	0.81±0.04	**0.95**±0.03	0.17±0.06	0.03±0.00	0.03±0.00	0.03±0.01	0.27±0.06
	ROC AUC	0.98±0.01	0.49±0.04	0.98±0.01	**0.99**±0.01	0.84±0.02	0.50±0.03	0.52±0.03	0.54±0.03	0.91±0.02
	P@n	**0.91**±0.04	0.02±0.02	0.76±0.04	0.90±0.03	0.22±0.07	0.02±0.02	0.02±0.02	0.02±0.02	0.32±0.05
Syn9	PRC AUC	**0.98**±0.02	0.02±0.00	0.92±0.03	0.96±0.03	0.21±0.03	0.03±0.00	0.03±0.00	0.03±0.00	0.36±0.04
	ROC AUC	**1.00**±0.00	0.49±0.05	**1.00**±0.00	**1.00**±0.00	0.88±0.02	0.50±0.04	0.51±0.05	0.56±0.04	0.94±0.02
	P@n	**0.93**±0.03	0.01±0.01	0.87±0.05	0.92±0.03	0.27±0.07	0.02±0.02	0.01±0.02	0.02±0.02	0.41±0.05
Syn10	PRC AUC	**1.00**±0.00	0.03±0.01	0.49±0.06	0.88±0.04	0.11±0.02	0.03±0.01	0.03±0.01	0.03±0.01	0.28±0.05
	ROC AUC	**1.00**±0.00	0.50±0.04	0.98±0.01	0.99±0.01	0.87±0.02	0.51±0.03	0.51±0.04	0.54±0.05	0.94±0.01
	P@n	**0.99**±0.01	0.04±0.03	0.45±0.05	0.78±0.05	0.12±0.04	0.02±0.02	0.03±0.02	0.03±0.01	0.32±0.04
Ranking[1]	PRC AUC	1.2	8.6	3.0	1.9	5.1	7.4	7.1	5.6	4.6
	ROC AUC	1.3	9.0	2.5	2.0	5.1	7.7	7.2	4.6	4.7
	P@n	1.1	8.0	3.2	2.0	5.0	7.3	7.3	5.9	4.3

[1] This is the average ranking of each anomaly detector on 10 synthetic datasets in terms of PRC AUC, ROC AUC and P@n, respectively.

QCAD is slightly worse than CAD and ROCOD in terms of PRC AUC. This demonstrates the effectiveness of QCAD in identifying contextual anomalies for different forms and degrees of dependencies between the behavioral features and contextual features. More importantly, the poor performance of LoPAD indicates the importance of distinguishing behavioral features from contextual features. Additionally, other traditional anomaly detectors—including IForest, LOF, k-NN, SOD, and HBOS—perform poorly on most datasets because they treat all features equally.

Another important observation is that QCAD is generally superior to other methods in terms of robustness. That is, QCAD attains high PRC AUC, ROC AUC, and Precision@n values with small standard deviations on Syn1, Syn2, Syn3, Syn4 and Syn5, indicating that it is robust to different forms and degrees of dependency relationships, including linearity and non-linearity. In contrast, its strongest contender, CAD, has high standard deviations on Syn1 and Syn6. One possible reason is that CAD gets trapped in a bad local minimum when using expectation-maximization algorithm to learn parameters. Despite being a contextual anomaly detector, ROCOD performs poorly on datasets Syn2 and Syn5 in terms of PRC AUC and Precision@n. From the results on Syn6, Syn7, Syn8, Syn9 and Syn10, it appears that a larger number of contextual features does not substantially affect the performance of QCAD. Compared to other contextual anomaly detectors, and specifically CAD, QCAD does not perform particularly better on these synthetic datasets.

The results on real-world datasets presented in Tables 4.6 and 4.7 show that QCAD performs best overall when compared to its contenders, as witnessed by its average ranking results. Concretely, QCAD outperforms the other methods on 13 out of 20 real-world datasets in terms of PRC AUC and ROC AUC, and on 11 out of 20 datasets in terms of P@n. On most of the remaining datasets, QCAD is on par with its strongest competitors. For instance, HBOS achieves the best performance on Forest Fires, Heart Failures, Hepatitis, and Indian Liver Patient; QCAD's performance on these datasets is comparable. The number of contextual features in these datasets is often substantially less than the number of behavioral features, reducing the contextual anomaly detection problem to a traditional anomaly detection problem. QCAD is only slightly worse than ROCOD on Power Plant and Toxicity in terms of PRC AUC. Note that CAD produces surprisingly good results on the Bodyfat dataset, and surpasses other methods including QCAD. One possible explanation is that the contexts and behaviors are both composed of well-separated Gaussian components, and that the association between contextual and behavioral components is strong.

We also observe that QCAD performs well on datasets with varying sample size,

Table 4.6: Performance in terms of PRC AUC, ROC AUC, and P@n, on real-world data, with 5 or 10 independent runs of injecting contextual anomalies into each dataset[1]. For each anomaly detector on each dataset, the mean value and standard deviation of each evaluation criterion is presented. The best results obtained on each dataset are highlighted in bold.

		QCAD	LoPAD	ROCOD	CAD[2]	IForest	LOF	k-NN	SOD	HBOS
Abalone	PRC AUC	**0.96**±0.01	0.02±0.00	0.55±0.04	0.27	0.39±0.05	0.92±0.01	0.94±0.00	0.75±0.02	0.19±0.03
	ROC AUC	**1.00**±0.00	0.36±0.05	0.98±0.00	0.93	0.98±0.00	**1.00**±0.00	**1.00**±0.00	0.96±0.01	0.91±0.01
	P@n	0.90±0.02	0.02±0.00	0.58±0.01	0.40	0.41±0.07	0.92±0.02	**0.93**±0.00	0.74±0.02	0.19±0.06
Airfoil	PRC AUC	**0.69**±0.05	0.05±0.01	0.41±0.05	0.46±0.04	0.10±0.02	0.05±0.01	0.05±0.01	0.14±0.02	0.09±0.02
	ROC AUC	**0.91**±0.03	0.51±0.03	0.79±0.02	0.76±0.03	0.72±0.03	0.53±0.04	0.52±0.02	0.78±0.02	0.66±0.02
	P@n	**0.66**±0.05	0.04±0.03	0.40±0.04	0.45±0.02	0.13±0.04	0.05±0.03	0.06±0.03	0.14±0.03	0.12±0.02
BodyFat	PRC AUC	0.60±0.11	0.08±0.02	0.54±0.00	**0.92**±0.08	0.16±0.03	0.09±0.02	0.09±0.02	0.10±0.04	0.18±0.04
	ROC AUC	0.91±0.05	0.48±0.04	0.50±0.00	**0.99**±0.00	0.73±0.04	0.51±0.06	0.52±0.05	0.53±0.08	0.76±0.04
	P@n	0.58±0.12	0.06±0.06	0.08±0.03	**0.88**±0.05	0.16±0.06	0.08±0.08	0.10±0.06	0.10±0.04	0.23±0.07
Boston	PRC AUC	**0.72**±0.05	0.08±0.01	0.30±0.06	0.33±0.13	0.11±0.02	0.08±0.01	0.08±0.02	0.08±0.01	0.13±0.03
	ROC AUC	**0.92**±0.03	0.48±0.03	0.83±0.04	0.70±0.11	0.60±0.04	0.48±0.04	0.49±0.04	0.51±0.04	0.66±0.04
	P@n	**0.68**±0.04	0.07±0.02	0.37±0.05	0.25±0.10	0.13±0.03	0.07±0.04	0.07±0.04	0.06±0.03	0.15±0.06
Concrete	PRC AUC	**0.63**±0.06	0.09±0.01	0.33±0.06	0.33±0.07	0.08±0.02	0.06±0.01	0.06±0.01	0.06±0.01	0.25±0.06
	ROC AUC	**0.93**±0.03	0.62±0.03	0.77±0.04	0.70±0.04	0.61±0.03	0.49±0.06	0.50±0.05	0.50±0.03	0.72±0.05
	P@n	**0.62**±0.05	0.1±0.03	0.33±0.04	0.33±0.06	0.08±0.03	0.06±0.03	0.06±0.02	0.05±0.02	0.29±0.04
El Nino	PRC AUC	**0.98**±0.01	0.11±0.01	0.40±0.02	0.57	0.53±0.08	0.01±0.00	0.01±0.00	0.04±0.00	0.77±0.03
	ROC AUC	**1.00**±0.00	0.91±0.01	0.98±0.01	0.91	0.97±0.01	0.50±0.02	0.50±0.03	0.90±0.00	0.99±0.00
	P@n	**0.93**±0.02	0.13±0.03	0.42±0.03	0.55	0.50±0.06	0.01±0.01	0.01±0.01	0.01±0.01	0.70±0.02
Energy	PRC AUC	**0.92**±0.02	0.05±0.01	0.42±0.06	0.32±0.05	0.35±0.07	0.10±0.03	0.37±0.02	0.89±0.04	0.31±0.05
	ROC AUC	**0.99**±0.00	0.40±0.05	0.82±0.03	0.62±0.04	0.88±0.03	0.55±0.04	0.95±0.00	0.98±0.00	0.79±0.02
	P@n	**0.81**±0.04	0.02±0.01	0.42±0.04	0.32±0.06	0.43±0.07	0.11±0.04	0.30±0.03	0.79±0.06	0.35±0.06
FishWeight	PRC AUC	**0.81**±0.05	0.13±0.06	0.46±0.10	0.41±0.12	0.23±0.07	0.12±0.04	0.12±0.04	0.11±0.03	0.21±0.06
	ROC AUC	**0.96**±0.02	0.52±0.05	0.82±0.07	0.69±0.11	0.77±0.06	0.51±0.10	0.52±0.09	0.56±0.08	0.69±0.06
	P@n	**0.71**±0.10	0.13±0.09	0.44±0.08	0.45±0.12	0.31±0.11	0.14±0.11	0.13±0.10	0.09±0.06	0.30±0.13
ForestFires	PRC AUC	0.99±0.00	0.10±0.01	0.48±0.05	0.29±0.07	0.97±0.02	0.17±0.02	0.18±0.02	0.58±0.04	**1.00**±0.00
	ROC AUC	**1.00**±0.00	0.50±0.04	0.88±0.02	0.85±0.05	**1.00**±0.00	0.71±0.04	0.74±0.01	0.96±0.01)	**1.00**±0.00
	P@n	0.96±0.02	0.09±0.04	0.41±0.04	0.28±0.12	0.94±0.02	0.16±0.05	0.12±0.04	0.60±0.04	**0.99**±0.01
GasEmission	PRC AUC	**0.94**±0.02	0.01±0.00	0.86±0.02	0.63	0.10±0.01	0.01±0.00	0.01±0.00	0.03±0.00	0.27±0.07
	ROC AUC	**1.00**±0.00	0.47±0.01	**1.00**±0.00	0.99	0.93±0.01	0.52±0.04	0.52±0.01	0.78±0.01)	0.97 ±0.00
	P@n	**0.89**±0.02	0.00±0.00	0.83±0.02	0.62	0.05±0.02	0.02±0.01	0.02±0.02	0.02±0.02	0.24±0.05
HeartFailure	PRC AUC	0.90±0.03	0.05±0.00	0.53±0.04	0.82±0.06	0.86±0.04	0.13±0.02	0.18±0.04	0.38±0.07	**0.97**±0.01
	ROC AUC	0.98±0.01	0.10±0.03	0.52±0.05	0.98±0.01	0.98±0.01	0.57±0.06	0.68±0.05	0.88±0.02	**1.00**±0.00
	P@n	0.83±0.05	0.00±0.00	0.16±0.14	0.78±0.05	0.80±0.05	0.12±0.06	0.20±0.07	0.34±0.07	**0.92**±0.03
Hepatitis	PRC AUC	0.98±0.01	0.05±0.01	0.55±0.03	0.79±0.07	0.92±0.03	0.17±0.02	0.14±0.02	0.35±0.01	**0.99**±0.01
	ROC AUC	**1.00**±0.00	0.49±0.06	0.98±0.00	0.99±0.00	**1.00**±0.00	0.87±0.02	0.84±0.01	0.96±0.00	**1.00**±0.00
	P@n	0.92±0.02	0.05±0.02	0.65±0.04	0.78±0.06	0.89±0.02	0.06±0.04	0.07±0.04	0.33±0.03	**0.96**±0.01

Table 4.7: (Table 4.6 Continued) Performance in terms of PRC AUC, ROC AUC, and P@n, on real-world data, with 5 or 10 independent runs of injecting contextual anomalies into each dataset[1]. For each anomaly detector on each dataset, the mean value and standard deviation of each evaluation criterion is presented. The best results obtained on each dataset are highlighted in bold.

		QCAD	LoPAD	ROCOD	CAD[2]	IForest	LOF	k-NN	SOD	HBOS
IndianLiver	PRC AUC	0.90±0.05	0.05±0.00	0.48±0.06	0.84±0.12	0.93±0.02	0.11±0.02	0.17±0.04	0.44±0.06	**0.99**±0.01
	ROC AUC	**1.00**±0.00	0.48±0.05	0.97±0.01	0.94±0.16	**1.00**±0.00	0.72±0.05	0.81±0.04	0.96±0.01	**1.00**±0.00
	P@n	0.85±0.03	0.05±0.02	0.47±0.07	0.74±0.25	0.90±0.02	0.07±0.03	0.20±0.10	0.40±0.07	**0.95**±0.02
Maintenance	PRC AUC	**0.91**±0.01	0.01±0.00	0.50±0.00	0.50	0.02±0.01	0.01±0.00	0.01±0.00	0.02±0.00	0.19±0.08
	ROC AUC	**1.00**±0.00	0.48±0.05	0.50±0.00	0.50	0.70±0.02	0.49±0.01	0.55±0.02	0.79±0.01	0.67±0.04
	P@n	**0.84**±0.02	0.01±0.02	0.04±0.01	0.00	0.03±0.02	0.00±0.00	0.01±0.00	0.01±0.01	0.32±0.09
Parkinson	PRC AUC	**0.81**±0.01	0.02±0.01	0.58±0.06	0.68	0.02±0.00	0.02±0.00	0.02±0.00	0.04±0.00	0.02±0.00
	ROC AUC	**0.99**±0.00	0.43±0.03	0.94±0.02	0.90	0.56±0.03	0.48±0.03	0.48±0.02	0.75±0.02	0.59±0.03
	P@n	**0.85**±0.01	0.02±0.01	0.54±0.05	0.62	0.01±0.01	0.04±0.01	0.06±0.01	0.02±0.01	0.01±0.01
PowerPlant	PRC AUC	0.56±0.03	0.02±0.00	**0.65**±0.05	0.32	0.04±0.00	0.01±0.00	0.01±0.00	0.01±0.00	0.22±0.03
	ROC AUC	**0.98**±0.01	0.45±0.01	0.97±0.01	0.64	0.78±0.03	0.50±0.03	0.49±0.02	0.63±0.03	0.71±0.04
	P@n	0.58±0.05	0.01±0.01	**0.68**±0.01	0.31	0.07±0.02	0.01±0.01	0.02±0.02	0.01±0.01	0.27±0.04
QSRanking	PRC AUC	**0.79**±0.04	0.06±0.00	0.57±0.08	0.24±0.07	0.09±0.03	0.09±0.03	0.09±0.02	0.09±0.01	0.47±0.08
	ROC AUC	**0.96**±0.01	0.37±0.02	0.89±0.03	0.74±0.04	0.48±0.07	0.48±0.07	0.48±0.04	0.51±0.04	0.87±0.03
	P@n	**0.72**±0.05	0.02±0.02	0.53±0.08	0.26±0.09	0.08±0.05	0.08±0.04	0.08±0.05	0.08±0.03	0.48±0.07
SynMachine	PRC AUC	**0.96**±0.03	0.34±0.04	0.79±0.06	0.42±0.07	0.34±0.05	0.80±0.04	0.91±0.00	0.71±0.06	0.26±0.03
	ROC AUC	0.98±0.01	0.78±0.03	0.89±0.03	0.65±0.07	0.84±0.03	0.92±0.03	**0.99**±0.01	0.85±0.03	0.68±0.02
	P@n	**0.94**±0.03	0.39±0.05	0.73±0.07	0.34±0.11	0.39±0.05	0.72±0.04	0.78±0.03	0.65±0.04	0.27±0.05
Toxicity	PRC AUC	**0.46**±0.09	0.10±0.01	**0.46**±0.06	0.50±0.06	0.10±0.01	0.08±0.02	0.07±0.01	0.12±0.02	0.10±0.01
	ROC AUC	**0.88**±0.03	0.57±0.03	0.84±0.03	0.56±0.13	0.71±0.04	0.60±0.04	0.60±0.03	0.73±0.06	0.69±0.03
	P@n	**0.49**±0.07	0.18±0.01	0.47±0.06	0.12±0.15	0.09±0.04	0.06±0.03	0.05±0.01	0.11±0.04	0.09±0.04
Yacht	PRC AUC	**0.90**±0.04	0.53±0.02	0.29±0.05	0.55±0.00	0.28±0.08	0.20±0.04	0.25±0.03	0.24±0.05	0.32±0.08
	ROC AUC	**0.98**±0.02	0.96±0.00	0.78±0.03	0.50±0.00	0.78±0.06	0.72±0.05	0.83±0.02	0.59±0.07	0.76±0.05
	P@n	**0.80**±0.06	0.58±0.06	0.30±0.06	0.08±0.04	0.32±0.08	0.19±0.05	0.24±0.04	0.25±0.05	0.30±0.07
Ranking [3]	PRC AUC	1.40	7.45	3.30	3.45	4.90	7.20	6.75	5.75	4.25
	ROC AUC	1.40	7.95	3.70	5.00	4.15	7.00	6.05	5.00	4.10
	P@n	1.45	7.35	3.65	4.00	4.60	6.95	6.25	6.05	4.10

[1] Due to computation time limitations, we perform 5 independent trials for datasets with a sample size greater than 2000. For smaller datasets, we conduct 10 independent trials.

[2] We only conduct an independent experiment using CAD on Maintenance, Gas Emission, Parkinson Telemonitoring, Power Plant, and Elnino. It would take more than 1 week for CAD to complete 5 independent trials on each of these individual datasets.

[3] This is the average ranking of each anomaly detector on 20 real-world datasets in terms of PRC AUC, ROC AUC, or P@n, respectively.

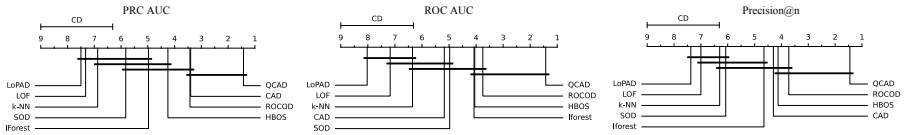

Figure 4.5: Critical difference diagram showing statistical difference comparisons between QCAD and its contenders in terms of PRC AUC, ROC AUC and Precision@n. To achieve this, we use Friedman tests [182] followed by Nemenyi post hoc analysis [183] with a significance level of 0.05. The post-hoc Nemenyi test indicates there are no significant differences within QCAD, CAD and ROCOD in terms of PRC AUC; there are no significant differences within QCAD, ROCOD, HBOS and IForest in terms of ROC AUC; there are no significant differences within QCAD, ROCOD, HBOS and CAD in terms of Precision@n. However, one can see that QCAD consistently outperforms its contenders by a large margin in three metrics.

dimensionality, and rate of injected anomalies. Specifically, QCAD achieved a high ROC AUC (≥ 0.85) on all datasets. This is much better than the ROC AUC values close to 0.50 obtained by ROCOD, CAD, HBOS, and IForest on some datasets, implying they are random guessing. In addition, QCAD attained high PRC AUC (≥ 0.8) and Precision@n values (≥ 0.7) on most datasets. The lowest PRC AUC and Precision@n value are is 0.46 and 0.49, respectively, both obtained on the Toxicity dataset. Possible reasons for QCAD's moderate performance on Airfoil, BodyFat, Power plant, and Toxicity are: the dependency relationship between behavioral features and contextual features is not strong, or the dependency relationship is too complex for the Quantile Regression Forests to capture. ROCOD, COD, HBOS, and IForest, however, obtain PRC AUC values less than 0.70 and Precision@n values less than 0.60 on most datasets. Moreover, HBOS and IForest sometimes attain PRC AUC and Precision@n lower than 0.10, which is extremely poor. This implies that traditional anomaly detection methods are not suitable for identifying contextual anomalies, as they treat all features equally. Furthermore, the relatively poor performance of ROCOD and CAD—compared to our method—demonstrates the importance of modeling more properties of the conditional distribution than just the mean.

4.6.5 Runtime Analysis

To investigate the scalability and efficiency of QCAD, we perform a runtime analysis by varying the sample size, the number of contextual features, and the number of behavioral features. As shown in Figure 4.6, we can empirically observe that the runtime of QCAD scales linearly with respect to the sample size, the number of contextual features, and the number of behavioral features on small and medium datasets. In

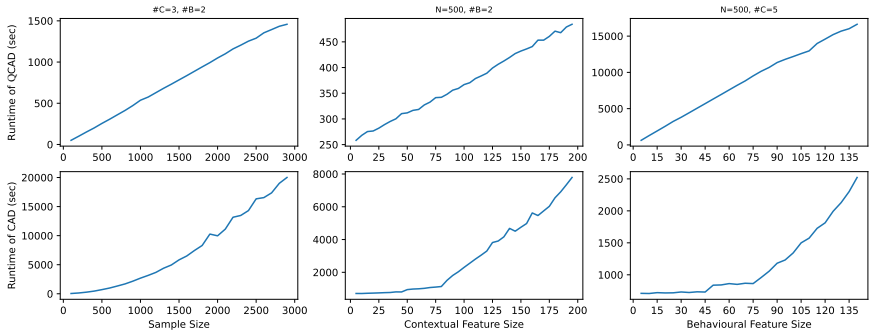

Figure 4.6: Runtime analysis by varying the sample size, the number of contextual features (#**C**), and the number of behavioral features (#**B**). The results are obtained with 5 independent trials. The data was synthesized using scheme S1. Note the varying y-axis scales.

contrast, CAD is computationally prohibitively expensive even on small datasets. To further understand the complexity of QCAD, we perform a time complexity analysis in Appendix 4.7. The theoretical analysis is in line with our empirical results and analysis.

4.6.6 Using QCAD to Find Promising Football Players

In this section, we illustrate the capability of QCAD on identifying potentially meaningful contextual anomalies and providing intuitive and understandable explanations for reported anomalies when applied to a real-world problem.

As use case, we consider the problem of finding exceptional players in the English Premier League. The dataset[3] describes the background information and performance statistics of 532 football players from 2020 to 2021. We use the variables *Position1,Position2* (positions for which the player plays), *Age* (age of the player), *Matches* (number of matches played), *Starts* (number of matches that the player was in the starting lineup), and *Mins* (number of minutes the player played overall) as contextual features. Two performance statistics, *Goals* (number of goals scored by the player) and *Assists* (number of assists given by player), are regarded as behavioral features.

Figure 4.7 depicts the distribution of all data objects in behavioral space *Goals* × *Assists*. Without considering contextual information, traditional anomaly detectors

[3]https://www.kaggle.com/rajatrc1705/english-premier-league202021

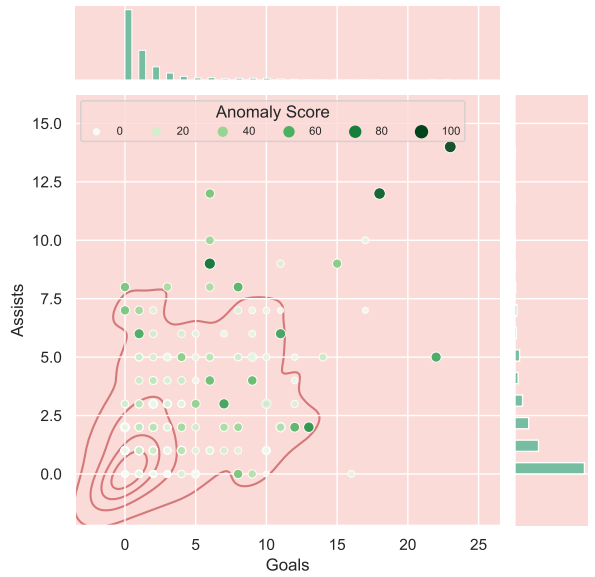

Figure 4.7: Distribution of all players in the behavioral space *Assists* × *Goals*, and their corresponding anomaly scores given by QCAD. The red curves represent the estimated densities of the joint distribution at different levels, the histograms indicate the marginal distributions. A larger and darker green dot represents a larger anomaly score as given by QCAD.

such as HBOS, LOF, IForest will treat objects in dense areas (i.e., objects residing inside the red curves) as normal objects. In contrast, our contextual anomaly detector QCAD can detect anomalies in dense areas by considering contextual information (i.e., green objects residing inside the red curves). Concretely, Figure 4.7 also presents the anomaly scores given by QCAD (with default settings). For example, player Matheus Pereira—who achieved 11 goals and 6 assists—is reported as an 'anomaly' (i.e., as having a relatively high anomaly score) by QCAD but as 'normal' by traditional anomaly detectors. Conversely, QCAD considers Patrick Bamford, a player with 17 goals and 7 assists (in a sparse area), to be normal, while traditional anomaly detectors consider him to be an anomaly.

In addition to giving an anomaly score for each data object, QCAD can also provide an explanation. For instance, for Matheus Pereira ($Position1$ = "MF", $Position2$ = "FW", Age=24, $Matches$=33, $Starts$=30, $Mins$=2577), Figure 4.8 shows the distribution of the values of each contextual feature for all players in his contextual neighborhood (i.e., similar players). It can be seen that most of these players are Midfielder (MF) and/or Forward(FW), aged from 22 to 28, playing matches more than

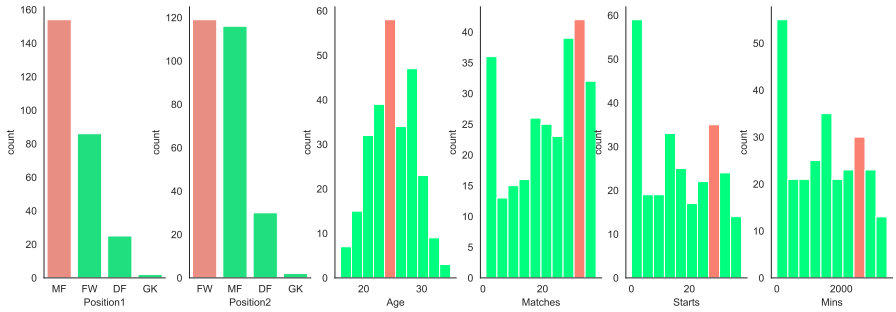

Figure 4.8: Explanations for the identified anomalous player Matheus Pereira based on his reference group. Shown are the values for Matheus Pereira (red bins) relative to the distribution of his contextual neighbors for each contextual feature.

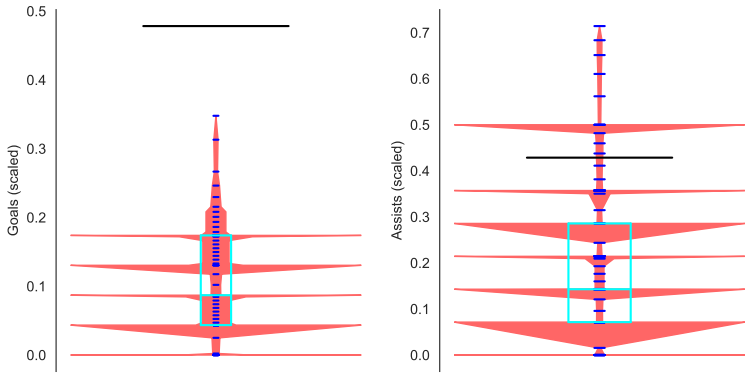

Figure 4.9: Explaining why Matheus Pereira is considered an anomaly: exceptionally many goals compared to contextually similar players, and relatively many assists. The beanplot shows the estimated conditional distribution of each behavioral feature, with a wider red area representing a higher probability of occurrence. The horizontal black lines represent the values reported for the player being investigated.

25 times, etc. The anomaly score is 65.8, which puts this player in the top-10 of most anomalous players. This score can be decomposed according to the behavioral features to give the behavioral feature(s) that contribute the most to the obtained anomaly score; *Goals*, in this case. Moreover, Figure 4.9 shows how the beanplot-like visualization can help to give insight in how the behavioral feature values deviate from those in the player's contextual neighborhood. From these plots we can see that Matheus Pereira has been exceptionally good at scoring goals and slightly better than expected with regard to giving assists.

These results can be interpreted as: Matheus Pereira performed surprisingly well in terms of goals and slightly better than expected in terms of assists when compared to other midfielders and/or forwards aged 22-28 who played more than 25 games and played more than 2000 minutes. Although we are not football experts, we expect the automated detection and interpretable score explanation of such 'anomalies' to be meaningful and possibly useful to football coaches and scouts.

4.7 Conclusions

In this paper, we for the first time explicitly establish a connection between dependency-based traditional anomaly detection methods and contextual anomaly detection methods. On this basis, we propose a novel approach to contextual anomaly detection and explanation. Specifically, we use Quantile Regression Forests to develop a accurate and interpretable anomaly detection method, QCAD, that explores dependencies between features. QCAD can handle tabular datasets with mixed contextual features and numerical behavioral features. Extensive experiment results on various synthetic and real-world datasets demonstrate that QCAD outperforms state-of-the-art anomaly detection methods in identifying contextual anomalies in terms of accuracy and interpretability.

From the case study on football player data, we conclude that QCAD can detect potentially meaningful and useful contextual anomalies that are directly interpretable by a domain expert. The beanplot-based visualizations help to explain why a certain object is (not) considered an anomaly within its context. This is important because anomaly detection is an unsupervised problem and the explanations can help analysts and domain experts to verify the results. It also opens up opportunities for human-guided anomaly detection, where feedback from the analyst can be used to guide the analysis.

In the future, given that QCAD can only handle static features, we plan to extend

QCAD to streaming settings.

Appendix A: Complexity Analysis

The computational overhead of our QCAD framework mainly comes from two parts: the calculation of Gower's Distance matrix using contextual features, and the calculation of anomaly score using all features based on Quantile Regression Forests.

The time complexity of calculating Gower's Distance matrix is $O(N^2 D_{cnt})$, where N represents the sample size and D_{cnt} denotes the number of contextual features. In addition, the time complexity of constructing a random forest is $O(n_{tree} \cdot n_{feature} \cdot klog(k))$ and using it for prediction is $O(n_{tree} \cdot n_{feature})$ [184], where n_{tree} is the number of trees used to form a random forest, $n_{feature}$ denotes the number of dimensions (i.e., D_{cnt} at most) and k indicates the number of samples used to construct the forest (i.e., the number of nearest neighbors in our case). Hence, the time complexity of constructing N quantile regression forests in D_{bhv} behavioral features is $O(n_{tree} \cdot D_{cnt} \cdot klog(k) \cdot N \cdot D_{bhv})$. Moreover, we have to estimate 100 different quantiles using each quantile regression forest, resulting in $O((n_{tree} \cdot D_{cnt} \cdot klog(k) + n_{tree} \cdot D_{cnt} \cdot 100) \cdot N \cdot D_{bhv})$. Therefore, using our default setting leads to $O((100 \cdot D_{cnt} \cdot \min(500, \frac{N}{2}) \cdot \log(\min(500, \frac{N}{2})) + 100 \cdot D_{cnt} \cdot 100) \cdot N \cdot D_{bhv})$. In the worst case, it is $O(10^5 N D_{cnt} D_{bhv})$.

Overall, the time complexity of our method is $O(10^5 N D_{cnt} D_{bhv} + N^2 D_{cnt})$ in the worst case. Particularly, the time complexity becomes $O(10^5 N D_{cnt} D_{bhv})$ when $N < 10^5$ (i.e., for small and medium datasets). Besides, the construction of quantile regression forests for each object in each behavioral feature can easily be performed in a parallel way, thus reducing the time complexity to $O(10^5 N D_{cnt} + N^2 D_{cnt})$ in the worst case.

Appendix B: Dataset Description

Abalone The dataset contains 4177 abalone physical measurement records. Specifically, we use the features Sex, Length, Diameter and Height as contextual features. Accordingly, we use the features Whole Weight, Shucked Weight, Viscera Weight, Shell Weight and Rings as behavioral features. This dataset is downloaded from UCI, [4] containing no missing values.

[4] https://archive.ics.uci.edu/ml/datasets/abalone

Airfoil Self-Noise This dataset contains 1503 records from a series of aerodynamic and acoustic tests of airfoil blade sections. Specifically, we use the features describing the frequency, the angle of attack, the chord length, the free-stream velocity and the suction side displacement thickness (e.g., f, alpha, c, U_infinity and delta) as contextual features. Meanwhile, the feature describing the scaled sound pressure level (e.g., SSPL) as behavioral feature. This dataset is downloaded from UCI, [5] containing no missing values.

Bodyfat This dataset contains the estimates of body density and the percentage of body fat (used as behavioral features), which are determined by various body circumference measurements, such as, Age, Weight, Height, Neck circumference, Chest circumference, Abdomen 2 circumference, Hip circumference, Thigh circumference, Knee circumference, Ankle circumference, Biceps (extended) circumference, Forearm circumference, and Wrist circumference (used as contextual features) for 252 men. The raw dataset includes no categorical features and 0 missing values, downloaded from the CMU statlib. [6]

Boston House Price This dataset contains 583 records of the Boston house price. We use the features describing the properties of the house and its surroundings, i.e., CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT as contextual features, and the median value of owner-occupied homes, i.e., MED, as behavioral feature. This dataset is released by [185] and downloaded from the CMU statlib [7]. It remains 506 records after dropping rows containing missing values.

Concrete Compressive Strength This dataset contains 1030 records about the compressive strength of concrete. We use the features describing the age and different ingredients as contextual features, including C1, C2, C3, C4, C5, C6, C7 and Age. In addition, we use the feature Strength as behavioral feature. This dataset is downloaded from UCI, [8] containing no missing values.

El Nino This dataset contains 178080 records of the oceanographic and surface meteorological readings, which are taken from buoys located throughout the equatorial Pacific. We use the temporal and spatial features, i.e., Year, Month, Day, Date,

[5] https://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise
[6] http://lib.stat.cmu.edu/datasets/bodyfat
[7] http://lib.stat.cmu.edu/datasets
[8] https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength

Latitude, Longitude as contextual features, and other features, i.e., Zonal_Winds, Meridional_Winds, Humidity, Air_Temp, Sea_Surface_Temp as behavioral features. This dataset is downloaded from the UCI machine learning repository.[9] It remains 93935 records after dropping rows containing missing values. However, in order for all algorithms to complete the experiment on this dataset within 12 hours, we randomly downsample the original dataset to 20,000 records.

Energy Efficiency This dataset contains 768 records about a study which assesses the energy efficiency as a function of building parameters. Accordingly, the building parameters such as X1, X2, X3, X4, X5, X6, X7 and X8 are regarded as contextual features, and the heating load and cooling load (namely Y1 and Y2) are treated as behavioral features. This dataset is downloaded from UCI, [10] containing no missing values.

Fish Weight This dataset contains 157 records about the features of common species of fish in the market. Accordingly, we use the features including Species, Length1, Length2, Length3, Height, Width as contextual features, and Weight as behavioral feature. This dataset is downloaded from Kaggle, [11] containing no missing values.

Forest Fires This dataset contains 517 records concerning meteorological and spatiotemporal information about forest fires in the northeast region of Portugal. We use the spatiotemporal features such as X, Y, month, day as contextual features, and the rest features, i.e., FFMC, DMC, DC, ISI, temp, RH, wind, rain, area as behavioral features. It is downloaded from the UCI machine learning repository,[12] containing no missing values.

Gas Turbine CO and NOx Emission The original dataset includes 36733 records of sensor measures data, which is collected from a gas turbine in Turkey from 2011 to 2015. The dataset is collected from the same power plant to study flue gas emissions and predict the hourly net energy yield. Consequently, we use the features describing the turbine parameters (e.g., AT, AP, AH, AFDP, GTEP, TIT, TAT and CDP) as

[9] archive.ics.uci.edu/ml/datasets/El+Nino
[10] https://archive.ics.uci.edu/ml/datasets/Energy+efficiency
[11] https://www.kaggle.com/aungpyaeap/fish-market
[12] https://archive.ics.uci.edu/ml/datasets/forest+fires

contextual features. The variables characterizing the turbine energy yield and emissions of gas (e.g., TEY, CO and NOX) are used as behavioral features. However, for all algorithms to complete the experiments within 12 hours, we only use the 7383 records in 2015. This dataset is downloaded from UCI, [13] containing no missing values.

Heart Failure This dataset contains heart failure clinical records of 299 patients suffering from heart failure. It contains 13 clinical features, of which we use age, sex, smoking, diabetes, high_blood_pressure, and anaemia as contextual features, and creatinine_phosphokinase, ejection_fraction, platelets, serum_creatinine, serum_sodium, time as behavioral features. Besides, the death_event feature is removed and there is no missing value in this dataset. The original dataset is released by [186] and we download it from the UCI machine learning repository. [14]

Hepatitis This dataset contains 615 records concerning the laboratory values of blood donors and Hepatitis C patients. We use the demographic features such as sex and age, and Category of donors as contextual features, and the rest of features, i.e., ALB, ALP, ALT, AST, BIL, CHE, CHOL, CREA, GGT and PROT as behavioral features. The original dataset is downloaded from the UCI machine learning Repository. [15] 26 records contain missing values and therefore are removed.

Indian Liver Patient This dataset contains the medical records of 583 Indian liver patients, 4 of which contain missing values and are therefore deleted from further analysis. We treat Age, Gender and Selector as contextual features, and Total_Bilirubin, Direct_Bilirubin, Alkaline_Phosphotase, Alamine_Aminotransferase, Aspartate_Aminotransferase, Total_Protiens, Albumin, Albumin_and_Globulin_Ratio as behavioral features. This dataset is downloaded from the UCI machine learning repository. [16]

Maintenance of Naval Propulsion Plants The data set contains 11934 experimental records, which were performed by a numerical simulator of a naval vessel featuring a gas turbine propulsion plant. We use the features describing the gas turbine measures of the physical asset as contextual features, including LeverPosition, GTT, GTn, GGn, Ts, Tp, T48, T1, T2, P48, P1, P2, Pexh, TIC, mf. Meanwhile,

[13] https://archive.ics.uci.edu/ml/datasets/Gas+Turbine+CO+and+NOx+Emission+Data+Set
[14] archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records
[15] https://archive.ics.uci.edu/ml/datasets/HCV+data
[16] https://archive.ics.uci.edu/ml/datasets/ILPD+(Indian+Liver+Patient+Dataset)

we use the features containing the ship speed and the performance decay over time of gas turbine components, e.g., ShipSpeed, CompressorDecay and TurbineDecay, as behavioral features. This dataset is downloaded from UCI, [17] containing no missing values.

Parkinsons Telemonitoring The data set contains 5875 records of a series of biomedical voice measurements from 42 early-stage Parkinson's disease patients. There people were recruited into a six-month trial of remote monitoring equipment for remote symptom progression monitoring. The features such as subject, age, sex, test_time, Jitter, Jitter_Abs, Jitter_RAP, Jitter_PPQ5, Jitter_DDP, Shimmer, Shimmer_dB, Shimmer_APQ3, Shimmer_APQ5, Shimmer_APQ11, Shimmer_DDA, NHR, HNR, RPDE, DFA, PPE descirbe the background information, and thus are used as contextual features. Meanwhile, the corresponding scores motor_UPDRS and total_UPDRS are used as behavioral features. This dataset is downloaded from UCI, [18] containing no missing values.

Power Plant This dataset contains 9568 records from a combined cycle power plant which worked with full load from 2006 to 2011. Features such as hourly average ambient variables temperature, ambient pressure, relative humidity and exhaust vacuum (e.g., T, AP, RH and EP) are considered as contextual features, while the net hourly electrical energy output (EP) is regarded as behavioral feature. This dataset is downloaded from UCI, [19] containing no missing values.

QS University Ranking This dataset contains the QS rankings of the world universities from 2018 to 2020. We consider the wolrd rankings, national rankings and location of each university from 2018 to 2020, i.e., World2018, National2018, World2019, National2019,World2020, National2020, Country as contextual features. We use six features that are considered for the ranking (i.e., Academic Reputation, Employer Reputation, Faculty to Student Ratio, Number of citations per faculty, International Faculty, International Students) as behavioral features. The original data is crawled from the QS website, [20] containing 475 records after dropping missing values.

[17] https://archive.ics.uci.edu/ml/datasets/Condition+Based+Maintenance+of+Naval+Propulsion+Plants
[18] https://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring
[19] https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant
[20] https://www.topuniversities.com/qs-world-university-rankings

QSAR Fish Toxicity This dataset contains 908 records about the fish Pimephales promelas. Specifically, the six features describing the molecular of 908 chemicals will be used as contextual features, and the corresponding acute aquatic toxicity measure will be used as behavioral feature. This dataset is downloaded from UCI, [21] containing no missing values.

Synchronous Machine This dataset contains 557 records from a real experimental set. This experiment aims to construct a model to estimate the excitation current of synchronous motors. Specifically, we use the features describing the load current, power factor, power factor error and changing of excitation current (e.g., Iy, PF, e and dIf) as contextual features. Accordingly, the feature which describes the excitation current of synchronous machine (namely If) is used as behavioral feature. This dataset is downloaded from UCI, [22] containing no missing values.

Yacht Hydrodynamics This dataset contains 308 records about the features of sailing yachts. We use the features describing the dimensions, velocity and hydrodynamic performance of yachts, namely Longitudinal_position, Prismatic_coefficient, Length_displacement_ratio, Beam_draught_ratio, Length_beam_ratio, Froude_number as contextual features, and the feature concerning the residuary resistance per unit weight of displacement, namely resistance, as behavioral feature. This dataset is downloaded from the UCI machine learning repository, [23] containing no missing values.

Appendix C: Parameter Sensitivity Analysis

The parameter k directly affects the effectiveness and efficiency of anomaly detector in anomaly detection phase. If our dataset is labeled, we can use techniques like grid-search and cross-validation to set an optimal k for a specific dataset. However, anomaly detection is usually an unsupervised learning problem, which makes it impossible to set an optimal k. Hence, we can only empirically give some rules of thumb to set k according to the properties of specific dataset (i.e., sample size, dimensionality and estimated rate of anomaly). Therefore, we perform intensive experiments on synthetic and real-world datasets with a wide range of sample sizes, dimensionalities and rates of injected anomalies to investigate the sensitivity of our algorithm on this parameter.

[21] https://archive.ics.uci.edu/ml/datasets/QSAR+fish+toxicity
[22] https://archive.ics.uci.edu/ml/datasets/Synchronous+Machine+Data+Set
[23] archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics

As shown in Figure 4.10, increasing the number of neighbors, i.e., k, will lead to a better performance in terms of PRC AUC, ROC AUC and Precision@n when k is small (about $N/10$ for most datasets). However, the performance gain gradually slows down as k increases, and the performance finally reaches a plateau with an increase of k. After that, further increasing k yields only a negligible performance gain, at the cost of runtime. Therefore, we set k to $N/2$ for small dataset and 500 for medium dataset after taking a trade-off between the accuracy and runtime cost. Overall, different from other k-NN based anomaly detectors which are sensitive to this parameter, our method QCAD is stable on parameter k as long as its value is not too small.

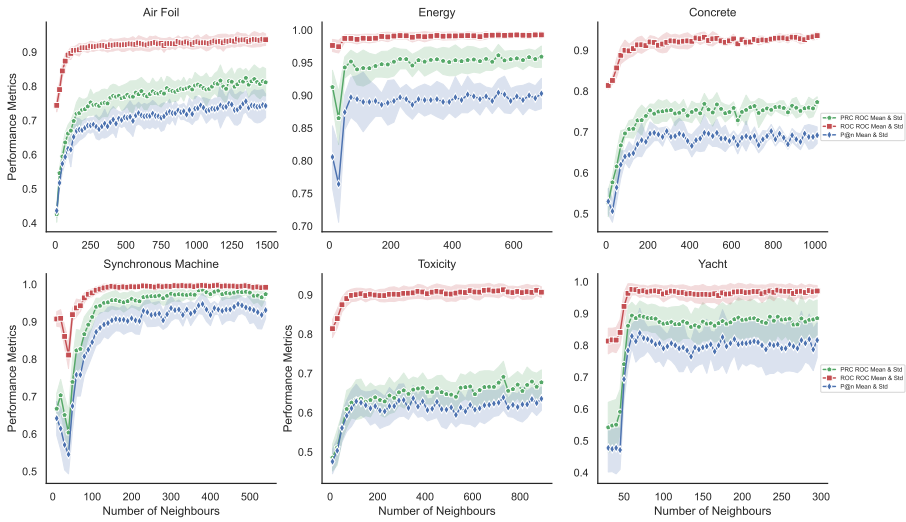

Figure 4.10: Sensitivity analysis on parameter k, where lines represent the mean values and the shaded areas indicate the corresponding standard deviations of each metric on 10 independent trials. Increasing the number of neighbors will first largely improve the performance in terms of PRC AUC (green line with asterisks), ROC AUC (red line with squares) and Precision@n (blue line with diamonds), and then yields negligible performance gains.

Appendix D: Ablation Study

4.7.1 Scaling Conditional Quantile Interval Length

In our method section, we have defined a matched conditional quantile interval length for an observation b^q when $b^q > \tau^q_{100}$ or $b^q < \tau^q_0$ as in Equation (4). We define the matched interval length for b^q based on $\max(w(\mathbf{x}|\mathbf{b}^q))$ and further scale it by

considering its distance to τ^q_{100} or τ^q_0. Alternatively, we can simply define the matched interval length as $\max(w(\mathbf{x}|\mathbf{b}^q))$ without scaling it. In other words, we could treat all observations residing outside $[\tau^q_0, \tau^q_{100}]$ equally. However, as shown in Figure 4.11, this will lead to a large performance degradation in terms of PRC AUC and Precision@n for most datasets.

Concretely, Figure 4.11 shows the difference of performance metrics, i.e., PRC ROC, ROC AUC and Precision@n between scaling $\max(w(\mathbf{x}|\mathbf{b}^q))$ with considering the distance to τ^q_{100} or τ^q_0, and not scaling $\max(w(\mathbf{x}|\mathbf{b}^q))$. For all datasets, the differences are positive. Particularly, these differences are significantly large in terms of PRC AUC and Precision@n for most datasets such as Energy, Hepatitis, Indian Liver Patient, Synthetic 5 and Synthetic 10. Therefore, it is pivotal to define the matched interval length by considering the distance from b^q to τ^q_{100} or τ^q_0 when $b^q > \tau^q_{100}$ or $b^q < \tau^q_0$, respectively.

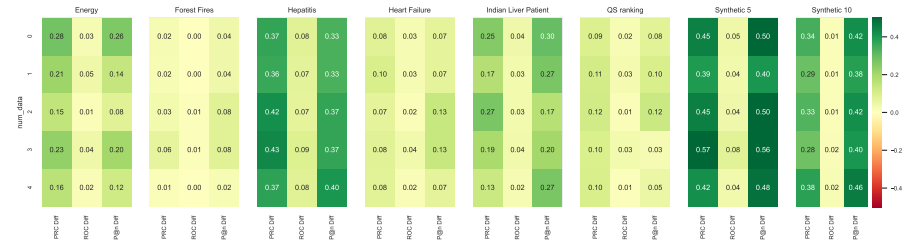

Figure 4.11: Effects of scaling matched conditional quantile interval length. For each dataset, the results are obtained by performing 5 independent trials of injecting contextual anomalies. For each trial, which is represented by the y-axis (namely *num_data*), the results are the difference (in terms of PRC AUC, ROC AUC, Precision@n, respectively) of methods with or without scaling the matched conditional quantile interval length. The large differences on most datasets imply the critical importance of scaling matched conditional quantile interval length.

4.7.2 Clipping Conditional Quantile Interval Length

To mitigate *dictator effect*, we have proposed to clip large matched conditional quantile interval lengths with an upper bound, which is set to $\frac{\eta}{100}$. More concretely, we set $\eta = 10$. To demonstrate the efficacy of this strategy on various datasets, we compute the performance metrics of QCAD by varying the hyper-parameter $\eta \in \{0.1, 0.2, 0.5, 1, 2, 3, ..., 11, 12, 15, 20, 30, 40, None\}$, where *None* means no clipping is performed.

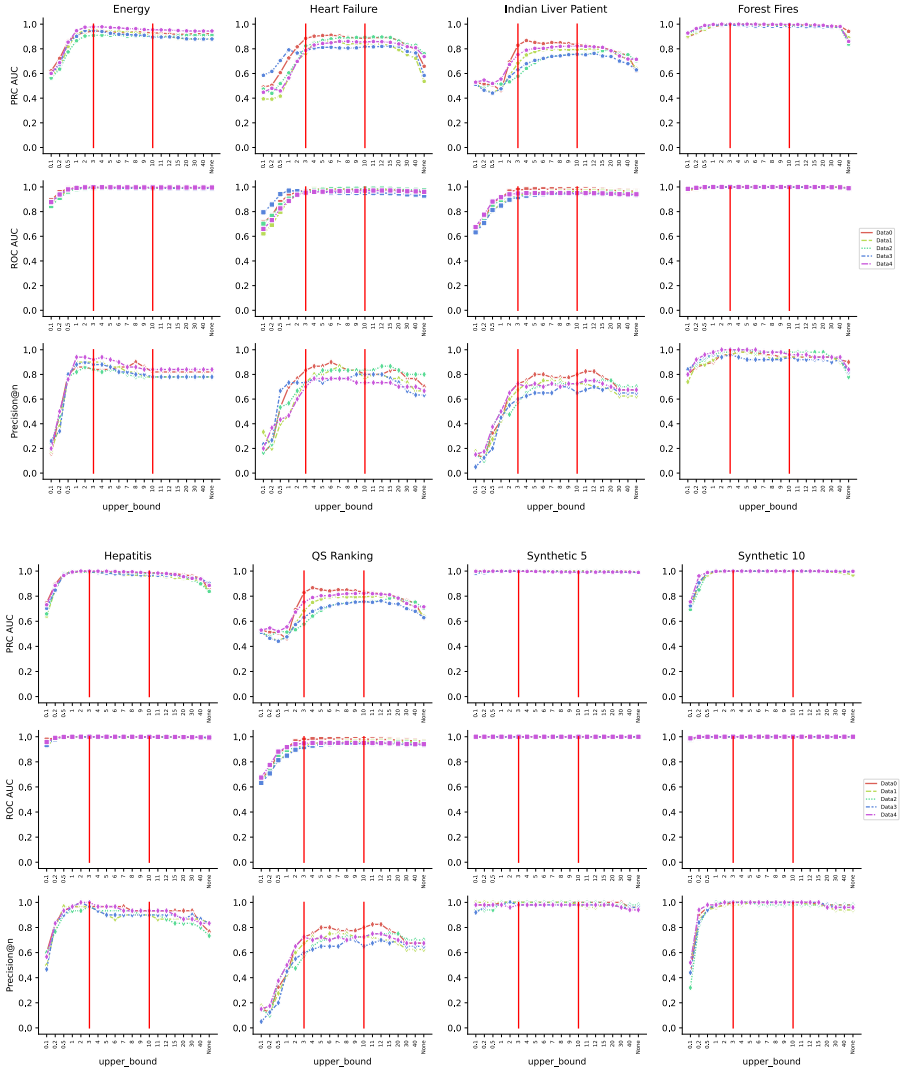

Figure 4.12: Effects of clipping matched conditional quantile interval length. For each dataset, the results are obtained by performing 5 independent trials (represented by 5 different curves) of injecting contextual anomalies. Then we compute the performance metrics (i.e., PRC AUC, ROC AUC, and Precision@n) of QCAD by varying the hyper-parameter η in $\{0.1, 0.2, 0.5, 1, 2, 3, ..., 11, 12, 15, 20, 30, 40, None\}$, where *None* means no clipping is performed. On most datasets, the best performances are achieved when $3 < \eta < 10$, as shown by the two vertical red lines.

As shown in Figure 4.12, increasing η will lead to higher PRC AUC, ROC AUC and Precision@n values when η is small (i.e., less than 1 for most datasets). Next, on the one hand, the ROC AUC stays stable as η increases, even without clipping (i.e. $\eta = None$). In other words, the clipping strategy does not affect ROC AUC metric. On the other hand, the PRC AUC and Precision@n gradually climb to a plateau as η increases. After a certain period, the PRC AUC and Precision@n start decreasing with an increase of η. Overall, on most datasets, PRC AUC and Precision@n generally achieve higher values with $3 \leq \eta \leq 10$ than those without clipping. Therefore, the *dictator effects* indeed exist and they mainly reduce the performance of QCAD in terms of PRC AUC and Precision@n. Moreover, our proposed clipping strategy can effectively overcome this problem.