The VLDB Journal (2024) 33:1203–1230
https://doi.org/10.1007/s00778-024-00857-w

SPECIAL ISSUE PAPER

Reliability evaluation of individual predictions: a data-centric approach

Nima Shahbazi[1] · **Abolfazl Asudeh**[1]

Received: 30 January 2023 / Revised: 5 March 2024 / Accepted: 10 May 2024 / Published online: 30 May 2024

Abstract

Machine learning models only provide probabilistic guarantees on the expected loss of random samples from the distribution represented by their training data. As a result, a model with high accuracy, may or may not be reliable for predicting an individual query point. To address this issue, XAI aims to provide explanations of individual predictions, while approaches such as conformal predictions, probabilistic predictions, and prediction intervals count on the model's certainty in its prediction to identify unreliable cases. Conversely, instead of relying on the model itself, we look for insights in the training data. That is, following the fact a model's performance is limited to the data it has been trained on, we ask "*is a model trained on a given data set, fit for making a specific prediction?*". Specifically, we argue that a model's prediction is not reliable if (i) there were not enough similar instances in the training set to the query point, and (ii) if there is a high fluctuation (uncertainty) in the vicinity of the query point in the training set. Using these two observations, we propose data-centric reliability measures for individual predictions and develop novel algorithms for efficient and effective computation of the reliability measures during inference time. The proposed algorithms learn the necessary components of the measures from the data itself and are sublinear, which makes them scalable to very large and multi-dimensional settings. Furthermore, an estimator is designed to enable no-data access during the inference time. We conduct extensive experiments using multiple real and synthetic data sets and different tasks, which reflect a consistent correlation between distrust values and model performance.

1 Introduction

Motivation: Interpretability is a necessity for data scientists who develop predictive models for critical decision-making. In such settings, it is important to provide additional means to support the following question: *is an individual prediction of the model reliable for decision-making?* To further motivate this, let us use Example 1:

Example 1 Using data-driven predictive models is prevalent for loan approval [73]. Consider a data science company that has developed a predictive model tailored for bankers, aiming to predict the chance of repayment by a prospective loan applicant. Indeed, such models can be beneficial to help financial institutes make wise loan application decisions. Suppose the model predicts the queried individual has a poor chance of repaying the loan in case of approval. The data science company and the bankers are aware and concerned about the critiques surrounding such models [11, 18, 56]. In particular, a major question the banker faces is whether or not they should rely on the prediction outcome to take action for this case. Therefore, the data science company would like to provide additional means alongside the model itself to help with the reliability question regarding individual predictions i.e. although the model demonstrates to be accurate on average, is it reliable for this individual prediction as well? Furthermore, for the cases in which the banker finds the model prediction unreliable, what evidence could be provided for them?

Novelty: There has been extensive research on trustworthy AI [39, 51, 58, 75, 80] to address the above issues. For example, explainable AI [30, 33, 66] provides simple explanations and rules that justify the model's prediction. On the other hand, approaches such as probabilistic predictions [59, 64, 84, 85], conformal predictions [7, 70], and prediction intervals [21, 40, 62] utilize the model's confidence in its prediction to identify instances deemed less reliable.

In contrast, rather than relying on the model itself, we seek insights within the training data used for drawing the prediction. Acknowledging that a model's accuracy is limited to the data on which it was trained, we pose the question: "*Is*

✉ Nima Shahbazi
nshahb3@uic.edu

Abolfazl Asudeh
asudeh@uic.edu

[1] University of Illinois Chicago, Chicago, IL, USA

a model trained on a specific dataset suitable for making a specific prediction?"

Technical highlights: We argue that, irrespective of the choice of model and its details, *the prediction for a query point is not reliable if the model is not trained on instances similar to the query point or if there is a high variance in the vicinity of the query point in the training set.* Therefore, we introduce two data-centric reliability measures based on the Representation and Uncertainty (RU) around the query point, called STRONGRU and WEAKRU. The RU measures are defined based on two components:

- *Representativeness:* Predictive models provide only probabilistic guarantees on the average loss over the distribution represented by the data set used for training them. As a result, their predictions are not distribution generalizable [44]. Consequently, if the query point is *not represented* by the data, the guarantees may not hold, hence one cannot rely on the prediction outcome.
- *Uncertainty:* When the query point belongs to an *uncertain* neighborhood with high variance on the target values, the model prediction may not be reliable.

WEAKRU is a warning that is raised if the individual prediction is problematic at least based on one of the two components. That is, if the query point belongs to an uncertain region or it is not represented by the data. STRONGRU, on the other hand, is a conservative but strong warning that is only raised when the query point fails based on both of the two components. That is, it both belongs to uncertain regions and is also not well-represented in the data. While WEAKRU is a weaker yellow flag (e.g., Fig. 1) warning that can be ignored for less critical decisions, STRONGRU is a red flag, which if raised, the model outcome should be ignored or at least considered with extra caution.

Example 1 (part 2): RU measures raise warning when the fitness of the data set used for drawing a prediction is questionable, helping the banker to be cautious when taking action. Besides, these measures provide quantitative justification to support the banker's action when they decide to ignore a prediction outcome that is not trustworthy. Suppose the WEAKRU for the query point is low. Besides the score, our system specifies that, for example, lack of representation is the issue, reflected by the low representation score. The banker can then argue to ignore a model outcome for this case, justifying that the model has been built using a data set that fails to represent the given case.

To see a concrete example, let us consider the mock interface in Fig. 1, showing an individual prediction with "poor chance or repayment". However, the ⟨`training data reliability`⟩ section raises a warning signal. Next, the ⟨`RU score breakdown`⟩ reveals that the model has not seen sufficient samples similar to this individual. That is,

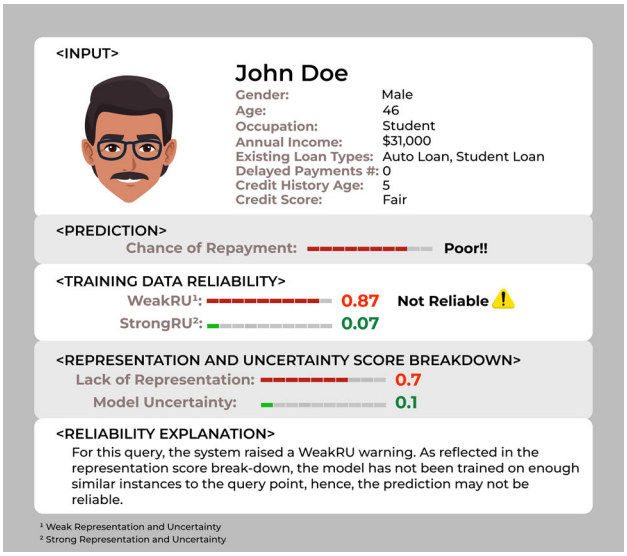


Fig. 1 Illustration of a mock interface for individual prediction reliability evaluation—Example 1 (Part 2)

the prediction suffers from a lack of representation. Based on this analysis, the banker can reject the model prediction and use the RU values along with the ⟨`reliability explanation`⟩ section to justify their action. □

While being agnostic to the choice of the uncertainty and lack of representation components, we propose an implementation based on the k-vicinity of a query point. In particular, given the radius of the k-vicinity and its uncertainty, we develop functions that return probabilities indicating the lack of representation and uncertainty. We propose methods to *learn* the probabilities *from the data set itself*. We devise proper indexing and algorithms that enable sublinear query processing that *scales* to large data sets.

Positioning in the context of existing work: Our work differs from existing literature including model-centric uncertainty quantification, local interpretation techniques, and data coverage in several radical ways:

- We offer a *data*-centric measure, a quantitative reliability warning that measures whether a query point is in the scope of use of a data. Unlike model-centric uncertainty quantification techniques, our techniques reveal a property of the data set that regardless of the constructed model this property stands still.
- Although model-centric techniques such as [21, 40, 62] guarantee a user-specified assurance level of error, this error is still computed over the entire data and consequently *may fail to focus the error on local regions* in data, representing, for example, minority populations in social applications. On the other hand, the local fidelity of our techniques satisfies equal treatment for every query point.

- While some model-centric uncertainty quantification techniques [40, 62] claim considering lack of representation as a source of uncertainty, as we observed in our experiments, they fail to capture the associated uncertainty for such query points in sparse regions. This failure originates from the perfect sampling assumption in development and production data which may not hold in practice. Our measures however properly capture such cases and directly target the lack of representation of a query point.
- The literature on data coverage [9, 10, 49] only focuses on representation, and hence fails to capture uncertainty. Additionally, they only return a binary signal of whether to trust the outcome of the model for a query point or not which practically is not very informative. Whereas our proposed measures target both sources of uncertainty and representation and return a quantitative value that is easily interpretable.
- Unlike techniques in interpretable machine learning [55] *justify* (advocate) individual predictions, our technique questions those that it finds unreliable.

Summary of contributions: In summary, our contributions in this paper include the following:

1. We propose data set RU measures to raise warnings when the fitness of a data set for an individual prediction is questionable. To the best of our knowledge, we are the first to propose data-centric RU measures, a property associated with data sets.
2. Our proposal is a quantitative measure based on two components: the query's lack of representation and uncertainty in the data set. The proposed measures can be extended to different data types and are independent of the model and prediction task (classification and regression). The measures are also agnostic to the choice of metric or approach for computing the two components. Proposing quantitative probabilistic outcomes, our measures are interpretable for the users since beyond the scores, the uncertainty and lack of representation components provide an explanation to justify them.
3. We propose novel algorithms based on the k-vicinity of a query point to compute the query's lack of representation and uncertainty. In particular, we "learn" the measurements from the data set itself. We also propose proper preprocessing and algorithms that enable sub-linear query answering that scales to very large and high-dimensional data sets. Furthermore, to enable no-data access during the query time, we build regression models to accurately estimate parameters needed to compute RU measures. We design an exponential search strategy for constructing large enough samples for training the estimators.
4. We conduct comprehensive experiments on multiple synthetic and real-world data sets with various scales and dimensions, on different prediction tasks (regression and binary/multi-class classification including text classification and image processing), using several models (such as Logistic Regression, DNN, Random Forest, etc.), and distance measures to (i) *validate* the effectiveness and consistency of the RU measures, (ii) evaluate the *efficiency and scalability* of our algorithms and (iii) evaluate the *existing works*.

Our extensive proof-of-concept experiments verify a consistent correlation between RU values and ML performance metrics on a variety of tasks, data sets, and ML algorithms. For tuples that have higher RU values (meaning they are less reliable w.r.t. to our measures), an ML model is more likely to fail to capture the truth and make a correct decision.

How to use? As demonstrated in our experiments, when RU values for a query point are high, one should discard or at least not rely on the individual prediction for critical decisions. We would like to reiterate that our proposal in this paper is complementary to the existing literature and *should be used alongside other techniques and potential approaches for trustworthy AI*.

2 Preliminaries

2.1 Data model

Consider a data set $\mathcal{D}$ with n tuples, each consisting of d (observation) attributes $\mathbf{x} = \langle x_1, x_2, \cdots, x_d \rangle$ and a target attribute y, also known as label attribute.[1] The observation attributes $\mathbf{x}$ are called the *input space* and the target attribute is called the *output space*. We assume the data set is used for training a prediction model h, as we shall further explain in Sect. 2.2. Prediction models assume that $\mathcal{D}$ is a set of iid (independent and identically distributed random) samples,[2] drawn from an (unknown) underlying distribution ξ. Attribute values may be discrete ordinal, continuous-valued, or non-ordinal categorical. Throughout the paper, we assume ordinal attributes are normalized in the range $[0, 1]$, with values drawn from the set of rational or real numbers. For non-ordinal attributes, we assume one-hot encoded repre-

[1] The measures and the algorithms proposed in this paper extend for data sets with multiple target attributes. In such cases, each measure is defined per each target attribute.

[2] We would like to note that our proposal does not make the iid assumption, which can be violated in practice, especially in the presence of issues such as sampling bias. We raise a warning when the data set is not fit to draw a specific prediction. As a result, in such cases, the warnings will be raised more frequently for the query points that are not represented by data.

sentations. We use $\mathbf{t}^j$ to refer to the j-th tuple in the data set $\mathcal{D}$ and its values of the observation attributes in particular. Similarly, we use y^j to refer to the value of the target attribute of $\mathbf{t}^j$. For every tuple $\mathbf{t} \in \mathcal{D}$, we use the notation t_i to show the value of $\mathbf{t}$ on attribute $x_i \in \mathbf{x}$.

2.2 Query and prediction model

The goal of prediction is to guess the target value y of a query point based on the observations on $\mathbf{x}$. In other words, given a point $\mathbf{q} = \langle q_1, q_2, \cdots, q_d \rangle$, the goal is to predict the value of the target attribute of $\mathbf{q}$. We consider the prediction model $h : \mathbb{R}^d \rightarrow \mathbb{R}$ as a function that predicts the target value of $\mathbf{q}$ as $h(\mathbf{q})$. When y is categorical, the task is classification, while regression is considered when y is continuous.

The underlying assumption is that $\mathbf{q}$ is drawn from the same distribution ξ from which $\mathcal{D}$ has been generated. Now, consider the Cartesian product of the input and output space $\mathbf{x} \times y$, and fix the hypothesis universe $\mathcal{H}$ of prediction functions. A learning algorithm A takes as input the set of samples in the data set $\mathcal{D}$ and finds a specific function $h = \mathsf{A}(\mathcal{D})$ by minimizing the empirical risk (maximizing the empirical accuracy or minimizing empirical loss) over $\mathcal{D}$. Empirical accuracy for classification is computed as the sum of samples in $\mathcal{D}$ for which the true label is the same as the predicted label:

$$\max \sum_{j=1}^{n} \mathbb{1}\Big(y^j == h(\mathbf{t}^j)\Big) \tag{1}$$

The equivalent objective for regression is to minimize the empirical error between the target variable and predicted values. Sum of Squares Error (SSE) is the de-facto error measure for regression:

$$\min \sum_{j=1}^{n} \Big(y^j - h(\mathbf{t}^j)\Big)^2 \tag{2}$$

Having a prediction model h trained by maximizing its empirical accuracy over the sample points in $\mathcal{D}$, the model is then used to predict the value of *unseen* target attribute of each query point $\mathbf{q}$, observed *after* model deployment, as $h(\mathbf{q})$. A central question at this point is whether a decision-maker should rely on the model prediction (at least for critical decisions). In the next section, we propose data-centric measures generated to answer this concern.

3 Reliability and uncertainty (RU) measures

Not every data set is fit for all data science tasks [9, 77]. An essential requirement for a learning algorithm is that its training data $\mathcal{D}$ should represent the underlying distribution ξ. Even if so, the trained model guarantees to perform well only *on average* over the query points drawn from ξ, not necessarily on a specific query point. To further explain this, let us provide some background from the machine learning theory.

Let $\mathcal{L}$ be the loss function used by the learning algorithm. Considering the underlying distribution ξ, the optimal model $h^* \in \mathcal{H}$ is the one with the minimum expected loss for a random sample drawn from ξ:

$$\mathcal{E}^*_{\mathcal{H}} = \inf_{h \in \mathcal{H}} \mathbb{E}\Big[\mathcal{L}\big(h(\mathbf{x}), y\big)\Big] = \inf_{h \in \mathcal{H}} \int_{\mathbf{x}} \mathcal{L}\big(h(\mathbf{x}), y\big) d_{\xi}(\mathbf{x}) \tag{3}$$

Let h_n be the model generated with the algorithm A over a data set $\mathcal{D}$ with n samples drawn from ξ. Given the values $\epsilon, \delta > 0$, the *sample complexity* [79] of A is the minimum value of n such that

$$\mathbb{P}_{\xi}\big(\mathcal{E}(h_n) - \mathcal{E}^* > \epsilon\big) \leq \delta \tag{4}$$

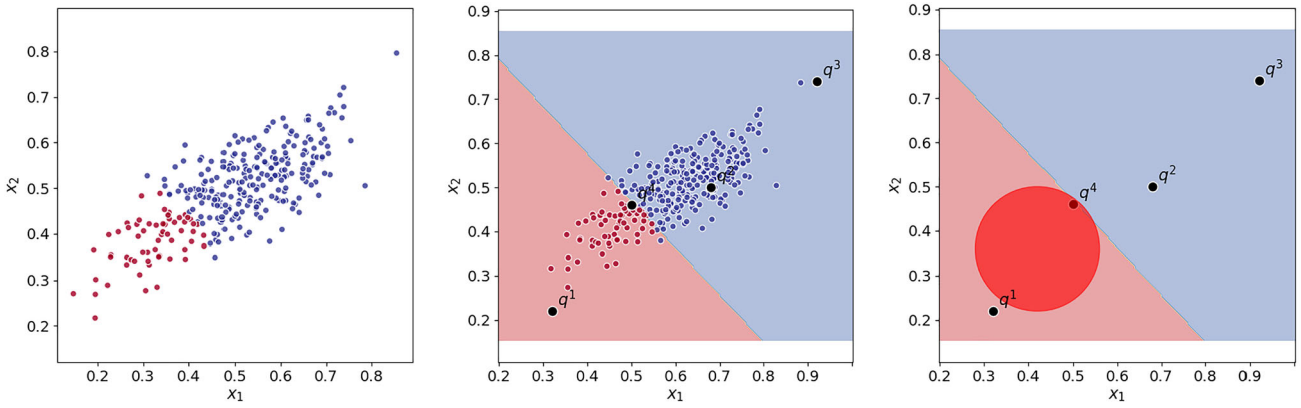
If the sample complexity of A for given values of ϵ and δ is unbounded, the function space is *not learnable*. The interesting immediate question is whether a distribution-free function is learnable. In other words, is there a learning algorithm A such that its sample complexity is bounded, independent of the underlying distribution? Unfortunately, following the so-called "no free lunch" theorem [37], the answer to the above question is negative.

In summary, a trained model h only provides probabilistic guarantee on the *expected* loss on random samples from the *underlying distribution* ξ represented by the data set $\mathcal{D}$. While ML models guarantee to perform well on average over the query points that follow ξ, our objective is *use-case base*, i.e., on *a single query point*—as opposed to the average performance of the model over a set of samples. A model that performs well on *majority* of samples drawn from ξ will have a high performance on average. Still, it does not necessarily mean it will perform well on the *minorities* and outlier points [10]. To further observe this, we present an example that leads to the design of our measures:

3.1 A toy example

As the running example in this section, let us consider the following classification task:

Example 2 Consider a binary classification task where the input space is $\mathbf{x} = \langle x_1, x_2 \rangle$ and the output space is the binary label y with values $\{-1\,(\text{red}), +1\,(\text{blue})\}$. Suppose the underlying data distribution ξ follows a 2D Gaussian, where x_1 and x_2 are positively correlated as shown in Fig. 2a. The figure shows the data set $\mathcal{D}$ drawn independently from the distribution ξ, along with their labels as their colors. Using $\mathcal{D}$,



(a) the data set $\mathcal{D}$ generated using a Gaussian distribution where x_1 and x_2 are positively correlated

(b) the decision boundary of learned model h and query points $\mathbf{q}^1$ to $\mathbf{q}^4$

(c) ground-truth boundary, overlaid on the model decision boundary and query points

Fig. 2 A toy example (Example 2) representing a binary classification task

the prediction model h is constructed as shown in Fig. 2b. The decision boundary is specified in the picture; while any point above the line is predicted as +1, a query point below it is labeled as -1. The classifier has been evaluated using a test set that is an iid sample set drawn from the underlying data set ξ. The accuracy on the test set is high (above 90%), and hence, the model gets deployed for predicting the outcome of unseen query points. We cherry-picked four query points, $\mathbf{q}^1$ to $\mathbf{q}^4$, that are also included in Fig. 2b. Using h for prediction, $h(\mathbf{q}^1) = -1$, $h(\mathbf{q}^2) = +1$, $h(\mathbf{q}^3) = +1$, and $h(\mathbf{q}^4) = -1$. Figure 2c adds the ground-truth boundary to the search space, revealing the true label of the query points: every point inside the red circle has the true label -1 while any point outside of it is $+1$. Looking at the figure, $y^1 = +1$ while the model predicted it as $h(\mathbf{q}^1) = -1$.

Let us take a closer look at the four query points in this example and their placement w.r.t. the tuples in $\mathcal{D}$ used for training h. $\mathbf{q}^2$ belongs to a *dense region* with many training tuples in $\mathcal{D}$ surrounding it. Besides, all of the tuples in its vicinity have the same label $y = +1$. As a result, one can expect that the model's outcome $h(\mathbf{q}^2) = +1$ should be a reliable prediction. Similar to $\mathbf{q}^2$, $\mathbf{q}^4$ also belongs to a dense region in $\mathcal{D}$; however, $\mathbf{q}^4$ belongs to an *uncertain region*, where some of the tuples in its vicinity have a label $y = +1$, and some others have the label $y = -1$. Considering the uncertainty in the vicinity of $\mathbf{q}^4$, one cannot confidently rely on the outcome of the model h. On the other hand, the neighbors of $\mathbf{q}^1$ (resp. $\mathbf{q}^3$) are not uncertain, all having the label $y = -1$ (resp. $y = +1$). However, the query points $\mathbf{q}^1$ and $\mathbf{q}^3$ are not well represented by $\mathcal{D}$, as those would be *outlier* w.r.t. $\mathcal{D}$. In other words, $\mathbf{q}^1$ and $\mathbf{q}^3$ are unlikely to be generated according to the underlying distribution ξ, represented by $\mathcal{D}$. As a result, following the no-free-lunch theorem, one cannot expect the outcome of model h to be reliable for these points. Note that, as we observed in our experiments, model-centric techniques such as prediction intervals and conformal prediction fail to detect $\mathbf{q}^1$ and $\mathbf{q}^3$ as not trustworthy.

Looking at the ground-truth boundaries in Fig. 2c, h luckily predicted the outcome for $\mathbf{q}^3$ correctly, but it was not fortunate to predict the y^1 correctly. Nevertheless, since the model has not reliably been trained for these outlier points, its outcome *may or may not* be accurate for these query points, hence is not trustworthy.

3.2 Strong and weak RU measures

From Example 2, we observe that the outcome of a model h, trained using a data set $\mathcal{D}$ is not reliable for a query point $\mathbf{q}$, if:

- *Lack of representation:* $\mathbf{q}$ is not well-presented by $\mathcal{D}$. In other words, $\mathbf{q}$ is an outlier w.r.t. the tuples in $\mathcal{D}$. In such cases, the model has not seen "enough" samples similar to $\mathbf{q}$ to reliably learn and predict the outcome of $\mathbf{q}$.
- *Lack of certainty:* $\mathbf{q}$ belongs to an uncertain region, where different tuples of $\mathcal{D}$ in the vicinity of $\mathbf{q}$ have different target values. In a classification context, that means the tuples have different labels (similar to $\mathbf{q}^4$ in Example 2). Similarly, in a regression setting, $\mathbf{q}$ belongs to a high-fluctuating area, where tuples in the vicinity of $\mathbf{q}$ have a wide range of values on the target variable.

We design the data-centric RU measures based on these two observations. In order to identify if a query suffers from uncertainty or lack of representation, one could use a deterministic approach using a fixed threshold. Then if the number of similar samples to (resp. label fluctuation in vicinity of) $\mathbf{q}$ is larger than the threshold it is considered as unrepresented

(resp. uncertain). This approach, however, would be misleading since two numbers close to the threshold could be treated very differently. Also, all points on each side of the threshold would be considered equally represented (resp., certain). Instead, we consider *a randomized approach*, widely popular in the literature, including [25]. That is, instead of using fixed thresholds, a Bernoulli variable (a biased coin) is used that assigns $\mathbf{q}$ as unrepresented (resp., uncertain) based on the number of samples similar to it (resp., its neighborhood uncertainty). We represent the probability of the Bernoulli variables for lack of representation or uncertainty components as $\mathbb{P}_o$ and $\mathbb{P}_u$, respectively.

Note that the two Bernoulli variables $\mathbb{P}_o$ and $\mathbb{P}_u$ are independent from each other. That simply follows the argument that after specifying the number of similar samples to $\mathbf{q}$ whether or not it should be considered as unrepresented does not depend on the uncertainty in the neighborhood of $\mathbf{q}$. We will further discuss this in Sect. 5.3. Before formally defining the RU measures, we would like to emphasize that our definitions are agnostic and independent from how $\mathbb{P}_o$ and $\mathbb{P}_u$ are computed. We still shall provide the details of how to compute these probabilities in Sect. 4.

Definition 1 (STRONGRU) The *strong* representation and uncertainty measure is a probabilistic measure that considers the outcome of a model for a query point $\mathbf{q}$ untrustworthy if $\mathbf{q}$ is not represented by $\mathcal{D}$ **and** it belongs to an uncertain region. Formally, the strong representation and uncertainty measure is:

$$\begin{aligned}\mathsf{SRU}(\mathbf{q}) &= \mathbb{P}\big((\mathbf{q} \text{ is outlier}) \wedge (\mathbf{q} \text{ is in uncertain region})\big)\\ &= \mathbb{P}_o(\mathbf{q}) \times \mathbb{P}_u(\mathbf{q}) \end{aligned} \tag{5}$$

STRONGRU raises the warning signal only when the query point fails on *both* conditions of being represented by $\mathcal{D}$ and not belonging to an uncertain region. For instance, in Example 2 none of the query points fail both on representation and on uncertainty; hence neither has a high STRONGRU score. On the other hand, a high STRONGRU score for a query point $\mathbf{q}$ *provides a strong warning signal* that one should perhaps reject the model outcome and not consider it for decision-making.

STRONGRU is a strong signal that raises warning only for the fearfully-concerning cases that fail both on representation and uncertainty. However, as observed in Example 2 a query points failing *at least* one of these conditions may also not be reliable, at least for critical decision making. We define the weak representation and uncertainty measures to raise a warning for such cases.

Definition 2 (WEAKRU) The *weak* representation and uncertainty measure is a probabilistic measure that considers the outcome of a model for a query point $\mathbf{q}$ untrustworthy if $\mathbf{q}$ is not represented by $\mathcal{D}$ **or** it belongs to an uncertain region. Formally, the weak representation and uncertainty measure is computed as follows:

$$\begin{aligned}\mathsf{WRU}(\mathbf{q}) &= \mathbb{P}\big((\mathbf{q} \text{ is outlier}) \vee (\mathbf{q} \text{ is in uncertain region})\big)\\ &= \mathbb{P}_o(\mathbf{q}) + \mathbb{P}_u(\mathbf{q}) - \mathbb{P}_o(\mathbf{q}) \times \mathbb{P}_u(\mathbf{q}) \end{aligned} \tag{6}$$

4 Implementation of the measures

4.1 Lack of representation oracle

The first component of the RU measures identifies if the data set $\mathcal{D}$ misses to represent the query point $\mathbf{q}$. The oracle returns the probabilistic measure $\mathbb{P}_o$, indicating if $\mathbf{q}$ is an outlier in $\mathcal{D}$. Different techniques have been proposed to identify the outliers and the anomalies [16, 20, 27, 50, 65] of a data set. The RU measures proposed in this paper are agnostic to the choice of the outlier detection technique, and alternative approaches that can compute $\mathbb{P}_o$ are equally applicable.

Still, in this section we provide a new approach for computing the probability $\mathbb{P}_o$, indicating if $\mathbf{q}$ is an outlier. In particular, we follow the existing work [10, 16, 27, 65] by considering the k nearest neighbors of $\mathbf{q}$ in $\mathcal{D}$ for studying if it is an outlier.

Given a distance metric Δ, let $\rho_{\mathbf{q}} = \Delta_k(\mathbf{q}, \mathcal{D})$ be the distance of the k-th nearest tuple in $\mathcal{D}$ to $\mathbf{q}$. Considering euclidean[3] distance measure for Δ, $\rho_{\mathbf{q}}$ is the radius of the k-vicinity of $\mathbf{q}$, the tight hyper-sphere (circle in 2D) centered at $\mathbf{q}$ that includes exactly k tuples from $\mathcal{D}$. For example, Fig. 3 shows the k-vicinity of the query points $\mathbf{q}^1$ to $\mathbf{q}^4$ in Example 2. It is easy to see that smaller values of $\rho_{\mathbf{q}}$ correspond to denser k-vicinities around $\mathbf{q}$, meaning that the data set $\mathcal{D}$ is more representative of the query point. We use this observation to develop the lack of representation component $\mathbb{P}_o$. That is, we consider the k-vicinity of $\mathbf{q}$ and the value of $\rho_{\mathbf{q}}$ to identify whether or not $\mathbf{q}$ is represented by $\mathcal{D}$.

In particular, we would like to develop the function $O : \mathbb{R} \rightarrow [0, 1]$ that given the value of $\rho_{\mathbf{q}}$ returns the probability $\mathbb{P}_o(\mathbf{q})$. That is, $\mathbb{P}_o(\mathbf{q}) = O(\Delta_k(\mathbf{q}, \mathcal{D}))$. The function O takes a distance value as the input and returns a probability indicating if the query point with that k-vicinity radius is not represented by $\mathcal{D}$. It is clear that as the distance values increase, the probability $\mathbb{P}_o$ should monotonically increase as well. However, translating the distances to the probabilities is unclear and may vary from one data set to another.

[3] Please note that while we use euclidean distance for the explanation and examples in the paper, our metrics and algorithms are agnostic to the choice of the distance measure, and those equally work for other ones. We evaluate the impact of the choice of the distance measure (using multiple well-known measures) in our experiments. Our experiments show consistent results across different measures.

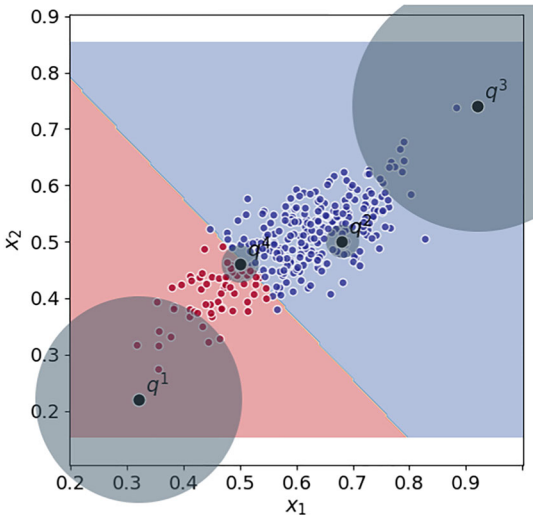


Fig. 3 Illustration of the k-vicinity ($k = 10$) of $\mathbf{q}^1$ to $\mathbf{q}^4$ in Example 2

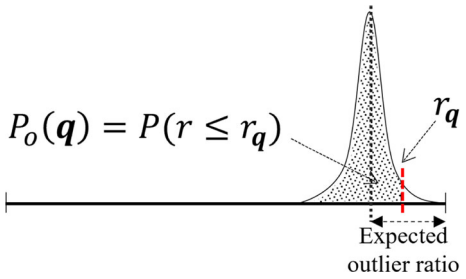


Fig. 4 Computation of of $\mathbb{P}_o(\mathbf{q})$ using the ratio of tuples in $\mathcal{D}$ with smaller k-vicinity radius than $\Delta_k(\mathbf{q}, \mathcal{D})$

Our idea is to *learn the function* O using the tuples in the data set $\mathcal{D}$. Specifically, we note that the probability of sampling an outlier tuple according to the underlying distribution ξ is low, and hence most of the tuples in $\mathcal{D}$ are not outliers. Therefore, the comparison between $\rho_{\mathbf{q}}$ and the k-vicinity radii of the tuples in $\mathcal{D}$ can reveal if $\mathbf{q}$ is an outlier. As a result, instead of directly translating the distance values to probabilities, we can first identify the *rank of* $\rho_{\mathbf{q}}$ in comparison with other tuples in $\mathcal{D}$ and use this information to specify if $\mathbf{q}$ is an outlier. For example, if $\rho_{\mathbf{q}}$ is smaller than more than half of k-vicinity radii of the tuples in $\mathcal{D}$, one can conclude that $\mathbf{q}$ is not an outlier. On the other hand, if $\rho_{\mathbf{q}}$ is larger than the k-vicinity radii of all tuples in $\mathcal{D}$, it should be an outlier.

Besides, it is often the case in practice that data sets are associated with information such as outlier ratio, showing approximately what percentage of its samples are outliers. We use such information to develop the function O.

In particular, since the ratio of the outliers in $\mathcal{D}$ is often an estimation by the experts, we consider a Normal distribution $\mathcal{N}(\mu, \sigma)$, where the user-specified outlier ratio is $(1-\mu)$ and σ is the standard deviation specifying the outlier ratio estimation variance. Figure 4 demonstrates such a distribution as

id	x_1	x_2	2-NN	ρ
t^1	0.61	0.58	$\{t^5, t^6\}$	0.191
t^2	0.32	0.77	$\{t^7, t^9\}$	0.161
t^3	0.79	0.41	$\{t^1, t^5\}$	0.247
t^4	0.13	0.9	$\{t^7, t^{10}\}$	0.082
t^5	0.74	0.44	$\{t^1, t^3\}$	0.191
t^6	0.55	0.64	$\{t^1, t^9\}$	0.183
t^7	0.18	0.85	$\{t^4, t^{10}\}$	0.147
t^8	0.93	0.12	$\{t^3, t^5\}$	0.372
t^9	0.38	0.71	$\{t^2, t^6\}$	0.183
t^{10}	0.05	0.92	$\{t^4, t^7\}$	0.147

Fig. 5 The data set in Example 3

a bell curve centered at one minus the expected outlier ratio. Recall that instead of directly using the value of k-vicinity radius to decide if a point is an outlier or not, we use the relative position of this value to compute the probability. That is, we define the probability distribution on the *ratio of outliers in* $\mathcal{D}$.

To do so, we first compute $\Gamma_{\mathcal{D}}$, the multi-set (including duplicate values) of k-vicinity radii of the tuples in $\mathcal{D}$. Now, let $r_{\mathbf{q}}$ be the percentage of values in $\Gamma_{\mathcal{D}}$ that are not larger than $\Delta_k(\mathbf{q}, \mathcal{D})$:

$$r_{\mathbf{q}} = \frac{|\{r \in \Gamma_{\mathcal{D}} | r \leq \Delta_k(\mathbf{q}, \mathcal{D})\}|}{n} \tag{7}$$

Using the value of $r_{\mathbf{q}}$, the query point $\mathbf{q}$ is an outlier if its k-vicinity radius falls within the range of outlier radii. In particular, suppose r is the boundary of outlier values in $\Gamma_{\mathcal{D}}$. Then $\mathbf{q}$ is an outlier if $r_q \geq r$. Following this argument, the function O can use the probability distribution $\mathcal{N}(\mu, \sigma)$ to compute the probability $\mathbb{P}_o(\mathbf{q})$. As shown in Fig. 4, $\mathbb{P}_o(\mathbf{q})$ is the probability that the outlier boundary r is less than or equal to $r_{\mathbf{q}}$, i.e. $\mathbb{P}_o(\mathbf{q}) = \mathbb{P}(r \leq r_{\mathbf{q}})$.

Converting the values to the standard-Normal distribution and using the Z-table:

$$\mathbb{P}_o(\mathbf{q}) = \mathbb{P}(r \leq r_{\mathbf{q}}) = \mathcal{Z}\left(\frac{r_{\mathbf{q}} - \mu}{\sigma}\right) \tag{8}$$

To further elaborate on how $\mathbb{P}_o(\mathbf{q})$ is computed, let us consider the following example:

Example 3 Consider the 2D data set $\mathcal{D}$ with $n = 10$ tuples shown in the table of Fig. 5. In addition to the tuple values on x_1 and x_2, the table also includes the k-NN ($k = 2$) of the tuples and the radius ρ of their k-vicinity. Let the outlier ratio of the data set be 20% ($\mu = 1 - 0.2 = 0.8$) with a standard deviation of $\sigma = 0.1$. Now consider the query point $\mathbf{q} : \langle 0.81, 0.76 \rangle$. The 2-NN of $\mathbf{q}$ are $\{t_1, t_6\}$, and $\rho_{\mathbf{q}} = 0.286$. Looking at the last column of Fig. 5, only t^8 has a larger k-vicinity radius than $\rho_{\mathbf{q}}$, i.e., for 90% of tuples the k-vicinity

radius is smaller than $\rho_{\mathbf{q}}$. Therefore, using Eq. 8, $\mathbb{P}_o(\mathbf{q}) = \mathcal{Z}((0.9 - 0.8)/0.1) = 0.84$.

Computing Eq. 8 requires (i) computing $\Delta_k(\mathbf{q}, \mathcal{D})$, which requires finding the k-NN of $\mathbf{q}$, and (ii) computing the value of $r_{\mathbf{q}}$. The baseline approach for computing these values makes a linear pass over $\mathcal{D}$ to identify the k-NN of $\mathbf{q}$. Besides, it requires $O(n^2)$ to compute the multi-set of k-vicinity radii $\Gamma_{\mathcal{D}}$ for the tuples in $\mathcal{D}$ and then it needs $O(n)$ to make a pass over $\Gamma_{\mathcal{D}}$ to compute $r_{\mathbf{q}}$.

However, given the interactive nature of query answering for ML systems and potentially large size of $\mathcal{D}$, we are interested in designing an algorithm that runs in a *sublinear* time to n. Theoretically speaking, finding the k nearest neighbors of a point can be done in $O(\log n)$ using k-voronoi diagrams[4] [4, 13, 22, 46], while constructing the k-voronoi cells takes polynomial time for a constant number of dimensions [4, 26] ($O(k^2 n \log n)$ for 2D [46]). Besides, practically efficient algorithms have been proposed [34, 76, 81], construct data structures in preprocessing time that enables identifying k-NN is near-logarithmic time. We rely on the off-the-shelf techniques for finding the k-NN of a query point.

During the preprocessing time, we first construct the k-NN data structure. Next, for every tuple in $\mathcal{D}$, we identify its k-vicinity radius and add it to the list $\Gamma_{\mathcal{D}}$. Finally, to quickly identify the value of $r_{\mathbf{q}}$ in query time, we sort the list Γ_D.

Algorithm 1

Input: data set $\mathcal{D}$; k; expected outlier ratio τ; standard deviation σ; query point $\mathbf{q}$

1: **function** PREPROCESS$_o(\mathcal{D},k)$
2: $M \leftarrow$ build the k-NN index of $\mathcal{D}$; $\Gamma \leftarrow [\,]$
3: **for** $t \in \mathcal{D}$ **do**
4: $V \leftarrow k$-NN(t)
5: add $\max_{t' \in V} \Delta(t, t')$ to Γ
6: **return** M, **sort**(Γ)
7: **function** $\mathbb{P}_o(\mathbf{q})$
8: $\rho_{\mathbf{q}} \leftarrow \max \Delta(\mathbf{q}, t'), \forall t' \in k$-NN$(\mathbf{q}, M)$
9: $r_{\mathbf{q}} \leftarrow \frac{1}{n}$BINARYSEARCH$(\rho_{\mathbf{q}}, \Gamma)$
10: **return** $\mathcal{Z}(\frac{r_{\mathbf{q}} - \mu}{\sigma})$

The preprocessing algorithm and the function for identifying $\mathbb{P}_o(\mathbf{q})$ are provided in Algorithm 1. To compute $r_{\mathbf{q}}$ for a query point $\mathbf{q}$, the algorithm first finds the k-vicinity of $\mathbf{q}$ and identifies the tuple in k-vicinity with maximum distance from $\mathbf{q}$. Next, it applies a binary search on the sorted list Γ to identify the number of cells in Γ that have a value not larger than $\mathbf{q}$, and use it to compute $r_{\mathbf{q}}$. At last, it uses Eq. 8 and returns the value of $\mathbb{P}_o(\mathbf{q})$.

Let T_{c_n} be the time to construct the k-NN index. Also, let $T_{q_n} \simeq O(\log n)$ be the time to identify the k-NN of a query point, using the constructed index. The preprocessing function constructs the k-NN index, identifies the k-NN of each tuple in $\mathcal{D}$, and finally spends $O(n \log n)$ to sort the list Γ. Therefore, the total preprocessing time is $O(T_{c_n} + n^2)$. Computing $\mathbb{P}_o(\mathbf{q})$ requires T_{q_n} to identify the k-NN of $\mathbf{q}$ and $O(\log n)$ for the binary search. As a result, the time to compute $\mathbb{P}_o(\mathbf{q})$ is $O(T_{q_n} + \log n) \simeq O(\log n)$.

[4] k-voronoi diagram is a partitioning of the query space into convex cells where the k-NN of all points in each cell is the same set of tuples.

4.2 Lack of certainty oracle

After the lack of representation oracle, we now turn our attention to the uncertainty oracle that, given the query point, the data set $\mathcal{D}$, and the target variable y, returns $\mathbb{P}_u$, the probabilistic measure that indicates if $\mathbf{q}$ belongs to an uncertain region. There has been extensive research, and there exist different metrics for computing uncertainty, namely, entropy, Gini impurity, Brier score, and probability calibration [15, 17, 29, 60, 71]. Indeed, we are agnostic to the choice of the technique for developing the uncertainty oracle, and any method that can compute the probabilistic measure $\mathbb{P}_u$ is equally applicable. Even so, in the rest of this section, we provide a development of the uncertainty oracle, following the technique proposed in Sect. 4.1. Similar to Sect. 4.1, we use the k-vicinity of a query point $\mathbf{q}$ as the region for studying uncertainty.

Binary classification is among the most popular ML tasks. A straightforward approach for developing the uncertainty oracle for such cases is to use the *Shannon entropy* ($\mathcal{H}$) [71]. Let $v_1, \cdots, v_\ell$ be the set of possible values for a target variable y. Known as a measure of uncertainty, the entropy of the random variable y is

$$\mathcal{H}(y) = -\sum_{i=1}^{\ell} \mathbb{P}(v_\ell) \log \mathbb{P}(v_\ell) \tag{9}$$

Higher entropy values refer to higher uncertainty, while values close to zero indicate a high certainty in the value of y. For a binary variable y, the maximum value of entropy is one, and it refers to the cases where the probability of each value is 0.5. Using entropy to measure uncertainty for binary classification, we consider the set of tuples $V_k(\mathbf{q}) \subseteq \mathcal{D}$ in the k-vicinity of the query point $\mathbf{q}$ and compute $\mathbb{P}_u(\mathbf{q})$ as the entropy among them. Let p_1 be the ratio of tuples in $V_k(\mathbf{q})$ with label 1. Then,

$$\mathbb{P}_u(\mathbf{q}) = -p_1 \log p_1 - (1 - p_1) \log(1 - p_1) \tag{10}$$

Entropy can also be used for non-binary classification. However, when the cardinality of y is larger than 2, entropy is not bounded by 1 anymore. Therefore, instead of using the absolute value of the entropy, we use the comparison between the uncertainty value of the query point vs. the uncertainty values in k-vicinities of the tuples in $\mathcal{D}$ to identify if $\mathbf{q}$ belongs

to an uncertain region. Intuitively, if the uncertainty in the neighborhood of $\mathbf{q}$ is smaller than a large portion of the other points in the training set, it is considered safe. Specifically, suppose r_u is the expected ratio of the tuples in $\mathcal{D}$ that belong to uncertain regions. Then, we consider a Normal distribution $\mathcal{N}(\mu_u, \sigma_u)$ where $\mu_u = (1-r_u)$ and σ_u is the standard deviation of uncertain ratio estimation. During the preprocessing, we construct $\Gamma_{u_\mathcal{D}}$, the sorted list of uncertainty values for the tuples in $\mathcal{D}$. Then, given a query point $\mathbf{q}$, we first compute $\mathcal{H}_\mathbf{q}(y)$, the uncertainty in the k-vicinity of $\mathbf{q}$ using Eq. 9. Next, applying a binary search on $\Gamma_{u_\mathcal{D}}$, we compute $r_{u_\mathbf{q}}$, the ratio of uncertainty values in $\Gamma_{u_\mathcal{D}}$ that are not larger than $r_{u_\mathbf{q}}$. Finally, converting the values to standard-Normal distribution, $\mathbb{P}_u(\mathbf{q})$ is computed as following:

$$\mathbb{P}_u(\mathbf{q}) = \mathbb{P}(r \leq r_{u_\mathbf{q}}) = \mathcal{Z}\Big(\frac{r_{u_\mathbf{q}} - \mu_u}{\sigma_u}\Big) \tag{11}$$

The residual sum of squares (RSS) is a popular measure for regression. In regression trees [15], for example, the objective is to split the search space into regions with high certainty, where the RSS values in each region are minimized, i.e., the certainty in each region is maximized. We also use RSS for measuring $\mathbb{P}_u(\mathbf{q})$ for the regression tasks. Let $V_k(\mathbf{q}) \subseteq \mathcal{D}$ be the set of tuples in the k-vicinity of $\mathbf{q}$. Also, let $m_{u_\mathbf{q}}$ be the average of y values in $V_k(\mathbf{q})$. That is, $m_{u_\mathbf{q}} = (\sum_{\mathbf{t}^i \in V_k(\mathbf{q})} y^i)/k$. Then the uncertainty around $\mathbf{q}$ is computed as

$$rss_\mathbf{q}(y) = \sum_{\mathbf{t}^i \in V_k(\mathbf{q})} \Big(y^i - m_{u_\mathbf{q}}\Big)^2 \tag{12}$$

The process for computing $\mathbb{P}_u(\mathbf{q})$ is the same as the one for classification, with the only difference being that RSS should be used for computing uncertainty (instead of entropy). The pseudo-code of the function $\mathbb{P}_u(\mathbf{q})$, along with preprocessing steps, are provided in Algorithm 2. Following a similar procedure as of Algorithm 1, the time to compute $\mathbb{P}_u(\mathbf{q})$ is $O(T_{q_n} + \log n) \simeq O(\log n)$. Using the functions $\mathbb{P}_o$ and $\mathbb{P}_u$, it takes a (near) logarithmic time to compute the uncertainty measures SRU and WRU.

4.3 No data access during the query time

During the query-answering phase, Algorithms 2 and 1 require to compute the k-vicinity radius and the entropy with the k-NN of a query point q. Although off-the-shelf k-NN indices are used to find this information, one could view it as requiring to access the training data after preprocessing. Our practical approach to address this is to *learn* these values. That is, to create two models that take a query point as the input, returning the k-vicinity radii and the entropy values. Creating these models requires sampling from the query space,[5] i.e., to generate a large-enough training set with observations being i.i.d samples from the query space, while the target variables are the k-vicinity radius and entropy. On the positive side, one can generate an arbitrarily large training set by generating i.i.d sample queries and then computing the target values using their k-NN. On the flip side, however, as proven in [10], the theoretical upper bound on the number of samples needed is exponential. In other words, theoretically speaking, the size of the training set may need to be exponential in the number of dimensions d for adversarial cases, in order to guarantee a given error ε. Fortunately, as we observe in our experiments, the theoretical upper bound is not tight, and in practice, the training set size is much smaller.

We still need to specify the proper training set size for our learning tasks. To do so, we design an *exponential search* algorithm as follows: the algorithm starts by setting the sample set size N_s to an initial value ($O(n)$). It then collects N_s i.i.d samples $\mathcal{S}$ from the query space and finds the k-vicinity of each sample s_i and identifies the k-vicinity radius[6] of s_i as $\rho_i \leftarrow \max \Delta(s_i, t'), \forall t' \in k\text{-NN}(s_i)$. Next, the algorithm builds a regression model $\mathcal{M}$ using $\mathcal{S}$ as the training set. After building the model, the algorithm checks if $\mathcal{M}$ has the error of at most ε, for a user-specified error ε. To check this, the model uses the test set $\mathcal{T}$. If $error > \varepsilon$, the algorithm *doubles the sample size* and repeats the process until it reaches the right sample size for N_s.

Algorithm 2

Input: data set $\mathcal{D}$; k; expected uncertainty ratio r_u; standard deviation σ_u; query point $\mathbf{q}$; k-NN index M

function UNCERTAINTY(V)
  if y is categorical /*`classification`*/ **then**
    $r_\ell \leftarrow \big|\{t^i \in V | y^i = v_\ell\}\big|/k,\ \forall v_\ell \in Dom(y)$
    return $-\sum_{i=1}^{\ell} r_\ell \log(r_\ell)$
  $m_u \leftarrow (\sum_{\mathbf{t}^i \in V} y^i)/k$
  return $\sum_{\mathbf{t}^i \in V} (y^i - m_u)^2$
function PREPROCESS$_u$($\mathcal{D}$,k)
  $\Gamma_u \leftarrow [\,]$
  for $t \in \mathcal{D}$ **do** add UNCERTAINTY(k-NN(t)) to Γ_u
  return sort(Γ_u)
function $\mathbb{P}_u(\mathbf{q})$
  $u_\mathbf{q} \leftarrow$UNCERTAINTY(k-NN($\mathbf{q}$, M))
  $r_\mathbf{q} \leftarrow$ BINARYSEARCH($u_\mathbf{q}$, Γ_u)/n
  return $\mathcal{Z}(\frac{r_\mathbf{q} - \mu_u}{\sigma_u})$
function SRU($\mathbf{q}$) **return** $\mathbb{P}_o(\mathbf{q}) \times \mathbb{P}_u(\mathbf{q})$
function WRU($\mathbf{q}$)
  $p_1 \leftarrow \mathbb{P}_o(\mathbf{q})$; $p_2 \leftarrow \mathbb{P}_u(\mathbf{q})$
  return $p_1 + p_2 - p_1 \times p_2$

[5] We refer to the space of valid queries as the query space. Specifically, let dom_i be the cardinality of the feature x_i. Then the query space is $\Pi_{i=1}^{n} dom_i$.

[6] The process to learn the entropy values is the same as learning the k-vicinity radii.

Let Reg_ρ and Reg_U be the trained regression models that return the k-vicinity radius and the k-NN entropy of a query point $\mathbf{q}$, respectively. Then, the only changes in the proposed algorithms are (i) replace Line 8 of Algorithm 1 with $\rho_\mathbf{q} \leftarrow Reg_\rho(\mathbf{q})$ and (ii) replace Line 8 of Algorithm 2 with $u_\mathbf{q} \leftarrow Reg_U(\mathbf{q})$.

5 Discussions

5.1 Assumptions and limitations

Let us begin our discussions by providing a synopsis of the assumptions underlying our approach's development and delve into the limitations it faces:

- RU measures are data-centric and and complementary to the the model-centric approaches for uncertainty quantification, such as conformal predictions, prediction intervals, and prediction probabilities, as well as the explainable AI literature. The RU measures should be considered alongside existing model-centric approaches in order to consider the influence of the model as well.
- Our data model defines a data set in a tabular manner, over a set of numeric attributes. For non-tabular data such as text, image, audio, etc., we rely on the existence of proper vector representations (aka embeddings), where each object is represented as a high-dimensional vector. We demonstrate this on an image data set in our experiments. The accuracy of our metrics is limited to the accuracy of the embeddings.
- In our development of lack of representation and uncertainty components, we assume the outlier ratio, the uncertainty ratio, and the variance values are provided as input, while considering a Normal distribution.
- To enable no-access to the training data at the inference time, we are required to learn the essential values via an exponential search across the query space with a $1-\varepsilon$ guarantee. This could become computationally expensive in the preprocessing step for small values of ε as a large number of samples are required.

In the remainder of this section, we present solutions aimed at tackling the limitations concerning hyper-parameters within our approach.

5.2 Parameter tuning

Similar to many other techniques in data mining and ML, the RU measures require parameter tuning. In implementing RU measures, we take the neighborhood size k in k-NN, along with the outlier ratio c of the training samples, the uncertainty ratio u, and the standard deviation for uncertainty and outlier distributions as hyper-parameters. The techniques proposed in this paper are agnostic to the choice of parameter tuning. Nevertheless, we present some heuristics for tuning these parameters in this section.

5.2.1 Tuning neighborhood size and outlier ratio parameters

The first parameter to determine is k: the number of tuples in $\mathcal{D}$ that specify the vicinity of the queried point. The second parameter is the outlier ratio c, which estimates the percentage of the tuples in the data set that are outliers.

To jointly tune c and k for a data set $\mathcal{D}$, we adopt a technique proposed in [82] for tuning the parameters of the local outlier factor (LOF) [16] algorithm. However, instead of choosing top $\lfloor cn \rfloor$ points with the highest LOF scores, we select top $\lfloor cn \rfloor$ with the highest k-vicinity radii.

We define a grid of values for c and k. For each combination, we calculate the k-vicinity radius for all tuples in $\mathcal{D}$, choose the top $\lfloor cn \rfloor$ tuples as the outliers, and the top $\lfloor cn \rfloor$ of remaining tuples as the inliers. The inliers are chosen in this manner because we are only interested in the tuples that are most similar to the outliers.

For each c and k, now we have a list of k-vicinity radii for outliers and a list for inliers and we calculate mean $(\mu_{out}(c,k), \mu_{in}(c,k))$ and variance $(\sigma^2_{out}(c,k), \sigma^2_{in}(c,k))$ over the log of the values in each list. We define the standardized difference in mean log k-vicinity radii between the outliers and the inliers as

$$T_{c,k} = \frac{\mu_{out}(c,k) - \mu_{in}(c,k)}{\sqrt{\lfloor cn \rfloor^{-1} \left(\sigma^2_{out}(c,k) + \sigma^2_{in}(c,k)\right)}}$$

If c is known, it is enough to find $k^*_c = \arg\max_k T_{c,k}$ that maximizes the standardized difference between the outliers and inliers for the corresponding c. Otherwise, we assume that k-vicinity radii form a random sample following a Normal distribution with the mean $\mu_{out}(c)$ and variance $\sigma^2_{out}(c)$ for outliers, and one with mean $\mu_{in}(c)$ and variance $\sigma^2_{in}(c)$ for the inliers. Then given a value of c, $T_{c,k}$ approximately follows a non-central t distribution with degrees of freedom $\mathsf{df}_c = 2\lfloor cn \rfloor - 2$ and the non-centrality parameter:

$$\mathsf{ncp}_c = \frac{\mu_{out}(c) - \mu_{in}(c)}{\sqrt{\lfloor cn \rfloor^{-1} \left(\sigma^2_{out}(c) + \sigma^2_{in}(c)\right)}}$$

We cannot directly compare the largest standardized difference T_{c,k^*_c} across different values of c because $T_{c,k}$ follows different non-central t distributions depending on c. Instead, we can compare the quantiles that correspond to T_{c,k^*_c} in each respective non-central distribution so that the comparison is on the same scale. To do so, we define $c_{opt} = \arg$

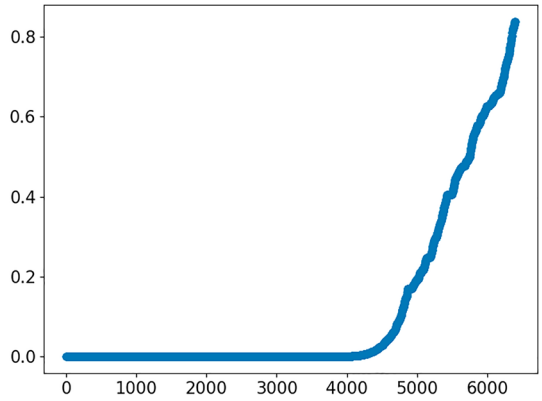


Fig. 6 Regression: sorted uncertainty values for *RN*

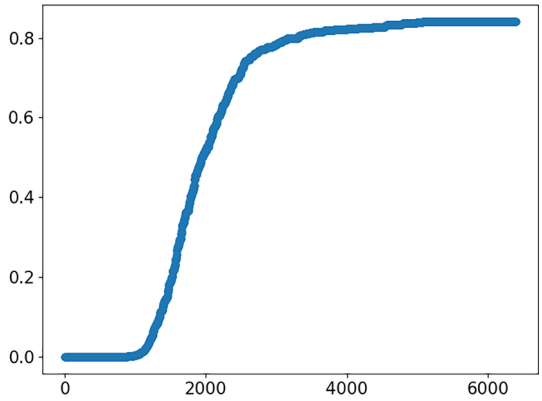


Fig. 7 Classification: sorted uncertainty values for multi-class classification data set shown in Fig. 11a

$\max_c P(z < T_{c,k_c^*}; \mathsf{df}_c; \mathsf{ncp}_c)$, where the random variable z follows a non-central t distribution with df_c degrees of freedom and ncp_c non-centrality parameter. c_{opt} is where T_{c,k_c^*} is the largest quantile in the corresponding t distribution as compared to the others.

5.2.2 Tuning uncertainty ratio parameter

The next parameter we need to tune is the uncertainty ratio u, which estimates what percentage of data belongs to uncertain regions. Similar to the outliers ratios that help us transform the k-vicinity radii to probabilities, the expected uncertainty ratio u helps us transform an uncertainty value in a k-vicinity to a probability. Consider the array $U_{\mathcal{D}} = \{u_1, \cdots, u_n\}$, where u_i is the uncertainty in the k-vicinity of $t_i \in \mathcal{D}$. We use the distribution of values $U_{\mathcal{D}}$ for identifying u. To explain the intuition behind this, let us consider a classification task. While the uncertainty for the tuples far from the decision boundary should be low, our experiments verify the uncertainty sharply increases as one gets close to the boundary. As a result, looking at the distribution of uncertainty values, one should be able to identify an estimation of u by finding the sharp slope in the distribution of uncertainty values. For example, Figs. 6 and 7 highlight our experiment results for two different settings for regression and classification. In Fig. 6, one can visually confirm that the sharp slope happens around 5000 with an uncertainty value of around 2. Similarly, in Fig. 7, the sharp slope happens around 2000, with an uncertainty around 0.4. Following this intuition, we calculate the k-vicinity uncertainty for each tuple in $\mathcal{D}$, and create the reverse cumulative distribution $\mathsf{V} : [0, 1] \rightarrow \mathbb{R}$ such that, for every value r, the ratio of tuples in $\mathcal{D}$ with an uncertainty value larger than $\mathsf{V}(r)$ is r. For example, $\mathsf{V}(0.1)$ returns the value $u_{0.1}$ such that the uncertainty for 10% of tuples is larger than it. We then identify the knee of this function (the sharp decrease in $\mathsf{V}(r)$) as the estimated uncertainty ratio. As a rule of thumb in our experiments, we observe that the knee falls around 10–15%.

5.3 Independence of RU measure components

Lack of representation and lack of certainty are the two components that are used in the definition of RU measures (Definitions 1 and 2). In particular, we use the radius of k-vicinity of the query point $\mathbf{q}$, $\Delta_k(\mathbf{q}, \mathcal{D})$, and define a Bernoulli random variable $\mathbb{P}_o$ on whether q is an outlier in $\mathcal{D}$. Similarly, we use the entropy in the vicinity of q to define the Bernoulli random variable $\mathbb{P}_u$ on whether $\mathbf{q}$ belongs to an uncertain region. Note that $\mathbb{P}_o$ only depends on $\Delta_k(\mathbf{q}, \mathcal{D})$ and does not change based on the entropy in the vicinity of $\mathbf{q}$.

Still, one can argue that in practice, there can be a correlation between uncertainty and sparsity of sub-spaces for a specific case. First, in our experiments, this correlation was very minimal, if not zero. Nevertheless, it is important to note that such correlation only impacts how frequently a similar pair of independent Bernoulli variables are sampled. To further clarify this, let us consider a toy example with a bag of paired coins, where the coins are biased. Suppose the paired coins have a positive correlation, i.e., if the first coin has a higher chance for the head, the other one has a high chance of being biased the same way. Now let us select a pair of coins from the bag. Suppose, the pairs are correlated, one has a probability of 0.8 and the other a probability of 0.75 for the head. Still, the pair of coins are independent of each other since the second coin will have a probability of 0.75 for the head, independent of the first coin. In our case, $\mathbb{P}_o$ and $\mathbb{P}_u$ can be modeled as paired independent coins, where their probabilities may (or may not) be correlated. However, after selecting a pair (by selecting a query point), the two variables are independent since whether a point is an outlier only depends on the density of points in its neighborhood not the variance in their target values (uncertainty).

6 Experiments

We conduct comprehensive experiments on multiple synthetic and real-world data sets of diverse sizes and dimensions using a variety of models (Logistic Regression, K-Nearest-Neighbor, Artificial Neural Networks, Deep Neural Networks, ElasticNet, Random Forest, and SVM), distance measures (Chebyshev, Manhattan, and Euclidean), and tasks (regression and binary/multi-class classification including text classification and image processing) to validate the effectiveness and consistency of our proposal and evaluate the efficiency and scalability of our algorithms. In our proof of concept experiments, we follow a similar evaluation approach to the ones conducted in the existing literature on the reliability of individual predictions [14, 63] where the correlation of the reliability scores and the prediction error over a test set is evaluated. We also demonstrate the failure of existing work such as Conformal Prediction, Prediction Probabilities (for cases that are not represented by the data), and data coverage (for the cases that belong to the uncertain regions) and how our proposed measures perform superior in capturing the prediction unreliability associated with point queries.

6.1 Experiments setup

The experiments were conducted using a 2.5 GHz Quad-Core Intel Core i7 processor, 16 GB memory, and running macOS. The algorithms were implemented in Python.

6.1.1 Motivating use-cases

We motivate our experiments based on the following example data science tasks:

- **Regression:** we consider three regression use-cases where (i) a GIS application requires to estimate the land altitude of a point $p = \langle long, lat \rangle$, (ii) a real-estate agency would like to predict house sale prices for investment, and (iii) a jewelry app would like to predict the price of diamonds based on their attributes. The RU measures will serve as warning signals when the altitude or price predictions are not reliable.
- **Classification:** We consider two classification use-cases where (i) a banking application that would like to predict the payment type of its credit-card holders and (ii) an employment application that needs to predict if an employee's salary is above 50K or not. In addition to class labels, the RU scores are provided as a reliability analysis of the predictions.
- **Text and image classification:** Last but not least, we consider two applications on text and image data as examples of unstructured data. In particular, we consider (i) classifying an aviation article and (ii) a handwritten digit recognition from images. Similar to the previous cases, we will use RU measures to identify when predicted labels are not reliable.

Considering these use cases for our non-synthetic experiments, the measure of success for RU measures is to see if the prediction reliability values are indeed aligned with these scores. That is, the evaluation is successful if the model predictions for queries with higher RU scores are worse with a higher probability. In other words, there should be a high correlation between the RU values and the model performance metrics.

We will use standard metrics for the model performances: Accuracy, F1, FNR, FPR for classification, and residual sum of squares (RSS) for regression.

6.1.2 Data sets

For evaluation purposes, we used (i) a collection of synthetic data sets and (ii) *seven real-world data sets* for regression and binary/multi-class classification including text and image classification.

Challenge: In the evaluation of our RU measures, we needed to generate samples with different RU values. However, since the tuples with high RU values are unlikely to be drawn from the underlying distribution ξ, it is challenging to collect enough samples (as a test set) to evaluate the effectiveness of our measures. A comprehensive evaluation requires query points drawn uniformly from the query space to cover different parts of it. To achieve this, we need to have access to a ground truth oracle that for *any* given sample taken from the query space returns the value of *target variable*. However, finding a real-world data set in a context where the ground truth oracle exists (publicly) is challenging. To overcome this challenge, we take three directions: first, to have full control of the shape and complexity of the ground truth labels over different data sets, we generate synthetic data; second, we find a real-world data set and a third party (public) service that provides access to ground truth labels; and third, we find a very large data set that contains samples from different parts of the query space and apply sub-sampling on it. Next, we remove the outliers from each sample (detecting the outliers using the Local Outlier Factor (LOF) algorithm) and split each cleaned sample into train and test sets, and add the outliers to the test set to cover larger parts of the query space. A downside of this approach is that it further reduces the presentation of points from under-represented regions in the training set, which may impact the model performance for those regions. Alternatively, one can partition the data in two halves and use one for the training set (without removing the outliers from it), while using the other solely for generating the test set. Indeed the training set size in such an approach

is smaller. We tried this approach in Sect. 6.6.3 and observed consistent results between the two approaches.

Real data sets: We use multiple real data sets, as briefly explained in the following:

1. **3D road network (RN) data set** [38] is a benchmark data set for **regression** that was constructed by adding elevation information to a 2D road network in North Jutland, Denmark. It includes 434,874 records with attributes `Latitude`, `Longitude`, and `Altitude`. We took 30 samples of size 10,000 from *RN* data set and generated 30 data sets and repeated each experiment 30 times, using different data sets. To address the evaluation challenge for the *RN* data set, we generated a uniform sample of 6,400 points $\langle x_1, x_2 \rangle$ in the range [0, 1]. We then transform the uniform samples back to the same space as the points in *RN*. To obtain the ground-truth labels for the query points, we used an *off-the-shelf API*[7] that given every coordinate ⟨`Latitude`,`Longitude` ⟩ in the data space, it yields the corresponding `Altitude`.
2. **House sales in King County (HS) data set** [32] is a **regression** data set for house sale prices for King County (Seattle). It includes houses sold between May 2014 and May 2015. It includes 21,614 records having 21 attributes with 2 categorical and 16 continuous types. Given attributes such as `no. of bedrooms`, `square footage`, `floors`, etc., the task is to predict the `price` of the house. We took 30 samples of size 10,000 from *HS* data set, generated 30 data sets, and repeated each experiment 30 times, using different data sets. To address the evaluation challenge for *HS* data set, for each sample, we removed the outliers and then split the data set into train and test sets. We then added the outliers back to the test set. Although with *HS* we can not measure the RU values for the whole query space, we believe the findings can still confirm the effectiveness of our measures.
3. **Diamond (DI) data set** [6] is a **regression** data set for predicting the price of diamond given some visual properties. This data set has 53,941 records with 14 attributes, 6 of which are continuous and 3 categorical. We used a similar approach to *HS* data set for utilizing *DI* in our experiments.
4. **Default of credit card clients (DCC) data set** [83] is a data set for **classification** that was constructed from payment data in October 2005 from an important bank in Taiwan. The data set is a binary class with default payment (Yes = 1, No = 0), as the response variable. Among the 30,000 records, 6,636 (22.12%) are cardholders with default payments. The data set has 23 features (9 categorical and 14 continuous) including `credit line`, `age`, `gender`, `education`, `history of payment`, `amount of bill statement`, `amount of the previous statement`, etc. Since it was not feasible for us to devise a function that can produce the ground truth for *DCC*, we took a sample of size 15,000 from the data set and then split it into two sets of train (5,000 tuples) and test (10,000 tuples) and used the test set as a substitute for the uniform sample over the query space. Following the same procedure, we generated 30 data sets and repeated each experiment 30 times, using different data sets. Similar to *HS*, we can not measure the RU values for the whole query space in *DCC*, yet the findings can still confirm the effectiveness of our measures.
5. **Adult (AD) data set** [42] is a well-known benchmark data set for **classification** tasks predicting whether income exceeds $50K annually based on census data. This data set has 32,561 records with 14 attributes, 6 of which are continuous and 8 categorical. We used a similar approach to the *HS* data set for utilizing *AD* in our experiments.
6. **Real-sim (RS) data set** [53] is based on *SRAA*[54] data set, preprocessed for SVMlin project [74]. The data set is designed for **text classification** tasks and is based on UseNet articles of four discussion groups on simulated auto racing, simulated aviation, real autos, and real aviation. The task is to separate real data from simulated data. *RS* is a sparse data set and has 72,309 records with 20,958 attributes with continuous values. We used a similar approach to *HS* data set for utilizing *RS* in our experiments.
7. **Gisette (GS) data set** [31] is a handwritten digit recognition data set based on the popular *MNIST* data set [45] for **image classification** to separate highly confusible digits '4' and '9'. The digits have been size-normalized and centered in a fixed-size image of 28 × 28 pixels. This data set has 6,000 records with 5,000 attributes with continuous values. We used a similar approach to *HS* data set for utilizing *GS* in our experiments.

Synthetic (SYN) data sets: To fully investigate the relationship between the RU measures and the model performance, we generated a collection of 60 data sets and repeated each experiment 60 times, using different data sets. Each data set is a random sample following a 2D Gaussian distribution with $\mu = [0, 0]$ and $\Sigma = \begin{bmatrix} 6 & 4 \\ 3 & 1 \end{bmatrix}$ over the input space $\mathbf{x} = \langle x_1, x_2 \rangle$ where x_1 and x_2 are positively correlated and the output space is the binary label y with values $\{-1, +1\}$. To create the binary classes for each data set, we randomly moved the samples over each shape in Fig. 8 in a way that the sample and shape have an intersection. As a result, each shape is the ground truth for 15 data sets but with differ-

[7] https://api.open-elevation.com/.

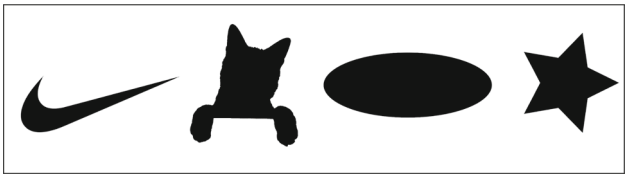


Fig. 8 Shapes used as the ground truth in creating the synthetic data sets *SYN*

ent placements. A data point belongs to the -1 class, if it falls into the corresponding shape, otherwise, it belongs to the $+1$ class. To address what we discussed in the evaluation challenge, we create a uniform sample of size 6,400 over [0, 1] and will label the points w.r.t. each shape and its placement in the space, generating a total of 60 uniform samples corresponding to each data set. In particular, following Example 2, we consider a binary classification task over the observation variables x_1 and x_2. We chose a 2D setting for visualization purposes. All continuous values used are normalized in the range [0, 1], using $(v_i - \min)/(\max - \min)$ and the non-ordinal ones are one-hot encoded using scikit-learn OneHotEncoder.

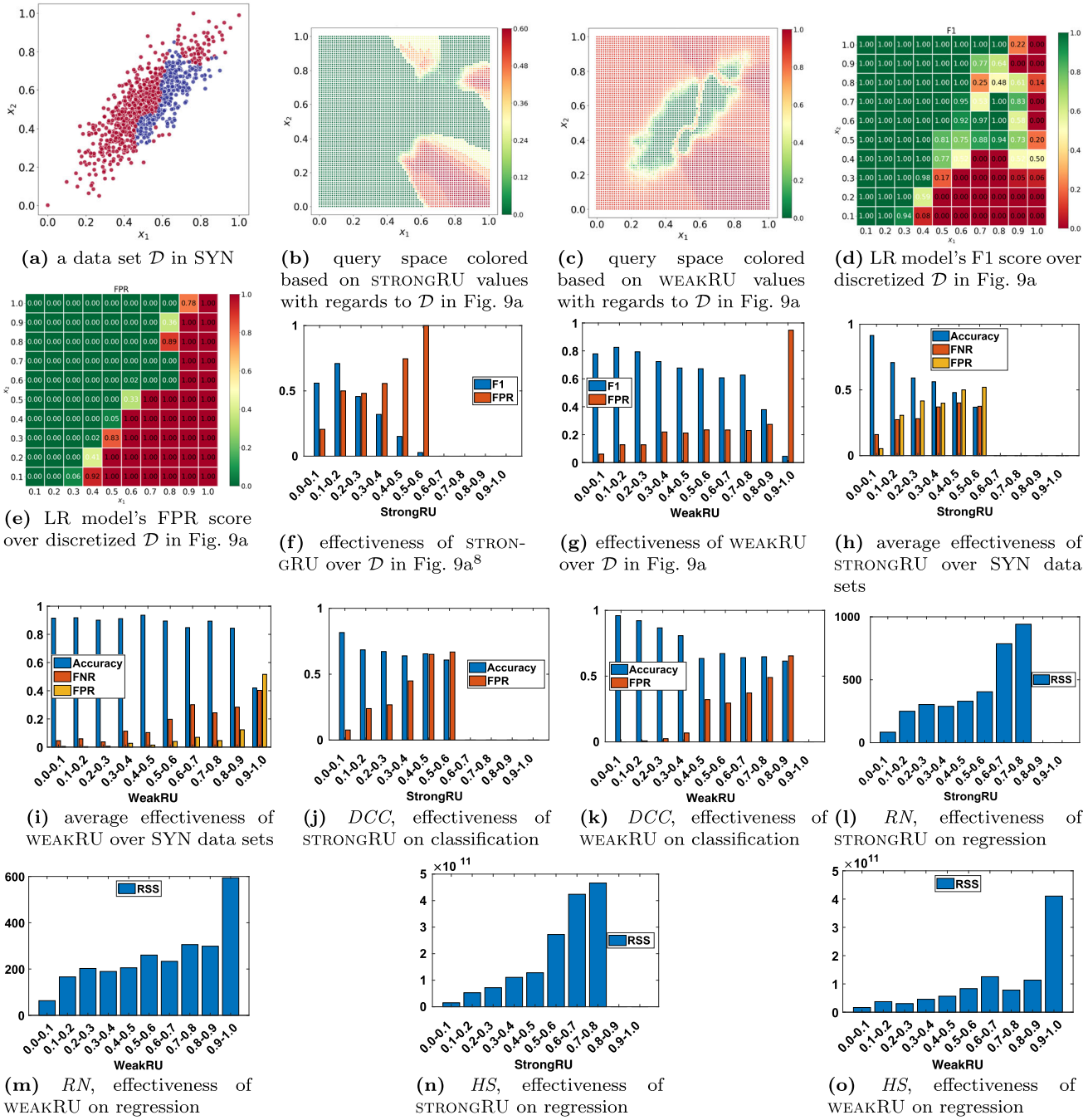
Default values: To evaluate the performance of our algorithms under different settings, we vary the value of a parameter, while fixing the value of the other ones. The parameters that are varied among our performance evaluation experiments include n (number of points), k (neighborhood size), c outlier ratio, and d number of dimensions. The default value for neighborhood size k is 10. The outlier ratio c is set to 0.1 suggesting that a mean $\mu = 0.9$ is chosen for outlier distribution with a standard deviation $\sigma = 0.1$. We adopt a technique proposed in [82] to jointly tune k and c parameters for a given data set. The tuning procedure for these two parameters alongside uncertainty ratio parameter u is discussed in detail in Sect. 5. The default value for d (number of attributes) is 2 for *SYN* and *RN* data sets, while it is 20, 18, 9, 14, 6,000, and 20,958 for *DCC*, *HS*, *DI*, *AD*, *GS* and *RS* respectively. The default value of n (size of data set) for the *SYN*, *RN*, *HS*, *DCC*, *DI*, *AD*, *GS* and *RS* data sets are 1,000, 10,000, 10,000, 5,000, 43,150, and 32,560, 6,000 and 72,309. The uncertainty ratio u is set to 0.1, therefore, a mean $\mu_u = 0.9$ is chosen for uncertainty distribution with a standard deviation $\sigma_u = 0.1$.

6.2 Proof of concept

We start our experiments by evaluating the effectiveness of RU measures across different data sets, ML models, and different parameters. Since the RU measures are model-independent, we perform the effectiveness validation experiments for both classification and regression tasks. For the classification tasks, we use *SYN*, *DCC*, *AD*, *RS* and *GS* data sets, and for the regression tasks, we employ *RN*, *HS* and *DI* data sets. To demonstrate the effectiveness of the RU measures we first provide a visual validation, using one of the 2D *SYN* data sets. We then present a comprehensive validation over all our data sets by providing the correlation between the RU values and the performance of an ML model's prediction on the same data. To do so, we deliver the results as bar graphs in which the x-axis is a bucketization of the ranges of the RU measures and the y-axis is the ML model's evaluation score. Each bar represents a value corresponding to a measure of accuracy/error i.e. *Accuracy, F1 score, FPR* and *FNR* of the ML model for all the tuples that have a RU value in the same range as the bar.

Visual validation: Consider the 2D data set $\mathcal{D}$ shown in Fig. 9a. $\mathcal{D}$ is borrowed from *SYN* as one of the 60 data sets with the shape of the cat as the ground truth (we obtained similar results for other data sets, as reflected in the aggregate values we shall report next). We compute STRONGRU and WEAKRU values for each query point in the uniform sample over the space using the default settings. In Fig. 9b and c, the query space is colored by assigning a tone based on the corresponding values of STRONGRU and WEAKRU respectively. As shown in Fig. 9b, the untrustworthy regions are the set of query points in the space that are both outliers w.r.t. the tuples in $\mathcal{D}$ **and** also uncertain since the entropy in their k-vicinity is high. On the other hand, in Fig. 9c, the untrustworthy regions are the set of query points that are either outliers **or** uncertain. The closer the color to red, the more untrustworthy the region will be and the opposite goes for green. Next, we train an arbitrary classifier (LR in this case) on data set $\mathcal{D}$ and evaluate the model's prediction. In this regard, we bucketize the uniform query space in a 10×10 grid and we evaluate the model for each cell to see where it falls short (*a.k.a.* model predictions become less reliable) then we create a heatmap on top of the grid based on the values for each cell. Results are provided in Fig. 9d and e. In a side-by-side comparison between the heatmaps and the colored space based on STRONGRU and WEAKRU (see Fig. 9b–e), it is easy to see that the model failed in the regions where the RU measures are producing large values. Finally, we generate the bar graphs with a similar procedure as later discussed in 6.2 and are shown in Fig. 9f and g. As the RU value increases, the F1 score of the model drops while the FPR rises meaning that, the model fails for the regions that are untrustworthy w.r.t. our RU measures. In Fig. 9f, as STRONGRU increases, the accuracy measures for the model rapidly drop, and for the 0.5−0.6 bucket, F1 is near zero. This confirms that while WEAKRU is a weaker warning, STRONGRU should be viewed as a red flag.

Validation on classification: Having provided the visual validation results, we next validate our RU measures on classification tasks. In this regard, using *SYN* data set, we first computed the RU measures for all the query points in the uni-



(a) a data set $\mathcal{D}$ in SYN

(b) query space colored based on StrongRU values with regards to $\mathcal{D}$ in Fig. 9a

(c) query space colored based on WeakRU values with regards to $\mathcal{D}$ in Fig. 9a

(d) LR model's F1 score over discretized $\mathcal{D}$ in Fig. 9a

(e) LR model's FPR score over discretized $\mathcal{D}$ in Fig. 9a

(f) effectiveness of StrongRU over $\mathcal{D}$ in Fig. 9a[8]

(g) effectiveness of WeakRU over $\mathcal{D}$ in Fig. 9a

(h) average effectiveness of StrongRU over SYN data sets

(i) average effectiveness of WeakRU over SYN data sets

(j) *DCC*, effectiveness of StrongRU on classification

(k) *DCC*, effectiveness of WeakRU on classification

(l) *RN*, effectiveness of StrongRU on regression

(m) *RN*, effectiveness of WeakRU on regression

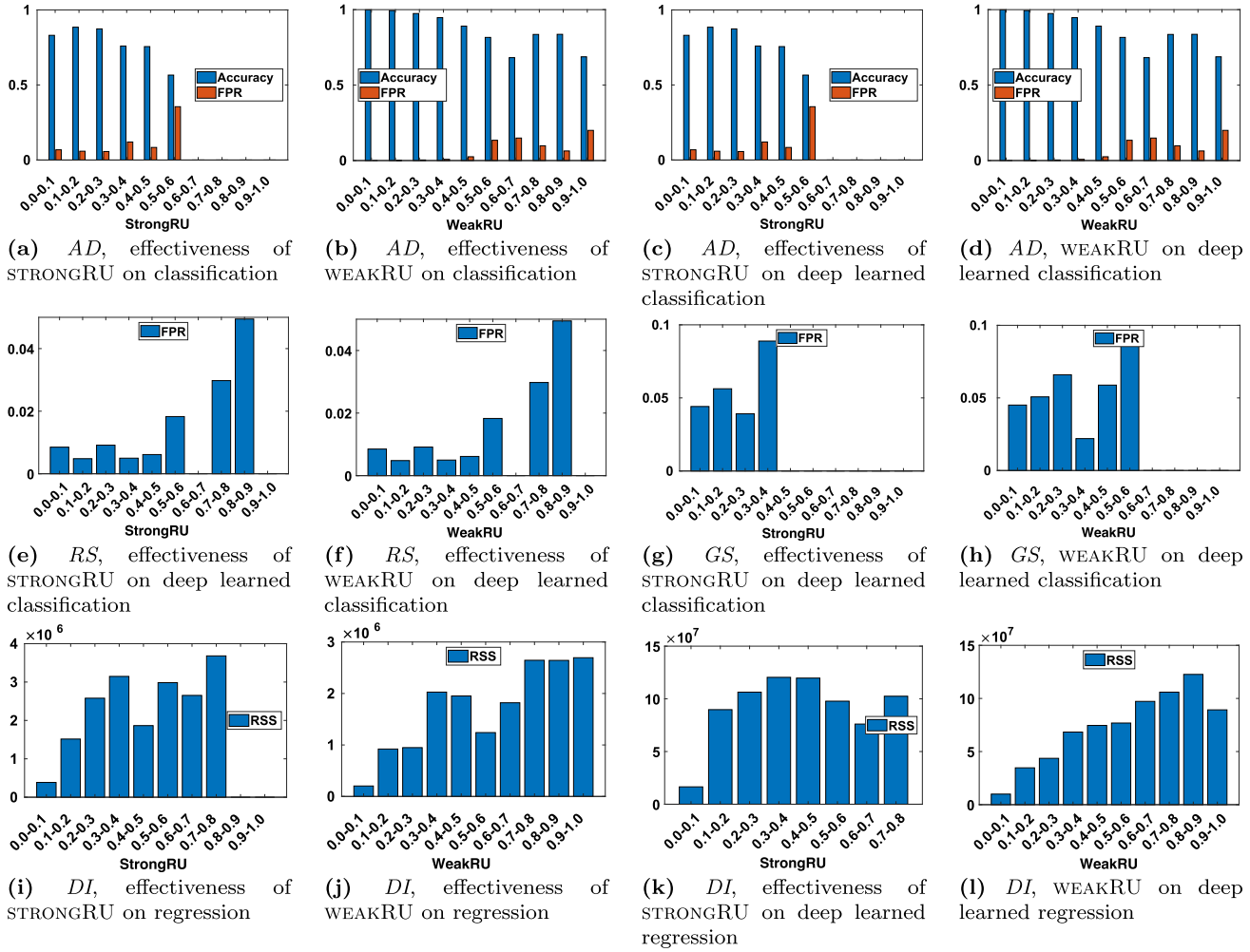
(n) *HS*, effectiveness of StrongRU on regression

(o) *HS*, effectiveness of WeakRU on regression

Fig. 9 Proof of concept results: consistent correlation between distrust values and ML performance metrics

form sample and bucketized the points w.r.t. their RU values in ranges of length 0.1. We repeated this for both StrongRU and WeakRU measures. Next, using a classification model that we trained on the (training) data set, we predict the target variable for the points in each range of RU measure. The values corresponding to the accuracy/error of the classifier over each bucket of RU values are provided in Fig. 9h and i for StrongRU and WeakRU respectively. As the RU values increase, the accuracy of the model drops while the FNR and FPR rise, and therefore, the model fails to capture the ground truth for the points that fall into untrustworthy regions in the data set.

To also perform the experiments on a real-world data set, we used the *DCC* data set with a similar procedure. The results are shown in Fig. 9j and k and they follow the same course as the previous experiment. We repeated the experiments with five classification algorithms including, Logistic Regression (LR), K-Nearest-Neighbor (k-NN),



(a) *AD*, effectiveness of STRONGRU on classification
(b) *AD*, effectiveness of WEAKRU on classification
(c) *AD*, effectiveness of STRONGRU on deep learned classification
(d) *AD*, WEAKRU on deep learned classification
(e) *RS*, effectiveness of STRONGRU on deep learned classification
(f) *RS*, effectiveness of WEAKRU on deep learned classification
(g) *GS*, effectiveness of STRONGRU on deep learned classification
(h) *GS*, WEAKRU on deep learned classification
(i) *DI*, effectiveness of STRONGRU on regression
(j) *DI*, effectiveness of WEAKRU on regression
(k) *DI*, effectiveness of STRONGRU on deep learned regression
(l) *DI*, WEAKRU on deep learned regression

Fig. 10 Additional proof of concept results on AD, DI, RS and GS datasets

Neural Networks (NN), Random Forest (RF), and SVM. All the classifiers underwent a hyperparameter tuning procedure to achieve optimal prediction correctness. We confirm that we obtained similar results in all cases and that all models failed to predict satisfactorily for query points that have a high RU value. Hence, we provide the results for the NN model, the more advanced and powerful model. Depending on the results of the classification, we chose the most appropriate accuracy/error measures that indicated the behavior of the model. For example, if the results were dominated by TPs and TNs, accuracy is the measure that best describes the model, otherwise, F1 might be a better choice. Depending on the number of FPs and FNs and the balance between them, the same rule applies to FPR and FNR measures.

Finally, to stress test our proposed measures with more complex learning tasks (such as complex classifiers, massive high dimensional data sets based on NLP and vision tasks, sparse data sets, etc.), we extended our experiments to three other classification data sets, *AD*, *RS*, and *GS*. Initially, we repeated the experiments with identical settings as before. For *AD*, the results are shown in Fig. 10a and b and corroborate our findings. For tuples with high RU values, models tend to fail more to predict reliably. Next, we repeated the experiments on *AD*, using deep learning. We trained a classification model (with tuned parameters) with two hidden layers of size 64 and 32 units respectively. We also constructed and tuned deep-learning models for *RS*, and *GS* data sets. The results are illustrated in Fig. 10c–h and are compliant with our previous experiments, showing that even with the choice of more complex models that show promising results in the tasks (95% F1 score in the case of *RS*), they still are less reliable for query points with high RU values. Models' accuracy in Fig. 10e–h were consistently above 95% in all cases. Therefore, for visual clarity we only included FPR.[8]

Validation on regression: In this experiment, we study the effectiveness of our RU measures in the regression tasks. Accordingly, we used *RN* and *HS* data sets and computed

[8] The absence of any records within certain RU ranges results in a 0 value for error/accuracy (y-axis) in those ranges.

Table 1 Pearson correlations between the RU values and model performance metrics for various datasets

dataset	pref. metric	dist. metric	Correlation
AD	Accuracy	STRONGRU	−0.88
	FPR	STRONGRU	0.84
	Accuracy	WEAKRU	−0.88
	FPR	WEAKRU	0.78
RN	MSE	STRONGRU	0.73
	MSE	WEAKRU	0.78
HS	MSE	STRONGRU	0.91
	MSE	WEAKRU	0.70
DI	MSE	STRONGRU	0.78
	MSE	WEAKRU	0.88

STRONGRU and WEAKRU values for all the query points in the uniform sample. Thereafter, we repeated the bucketization process as we did in the last experiment, and having trained a regression model over the data set, we evaluated the model's prediction over the tuples from each bucket. The results are presented in Fig. 9l–o. As the RU value increases, the RSS of the regression model follows the same trend denoting that the model fails to perform for tuples with a high RU value. We repeated the experiments with 3 different regression algorithms including ElasticNet, DT, and k-NN, all three with tuned hyper-parameters. Regardless of the regression model, the outcome was similar and therefore we only report the results for the k-NN regressor.

Finally, using the *DI* data set, we repeated the experiments with identical settings as before. The results are brought in Fig. 10i and j verifying our findings. For tuples with high RU values, models fail more frequently. Next, we repeated the experiments on *DI*, using a Deep Learning regression model with tuned parameters. We constructed a regression model with four hidden layers of size 128, 64, 32, and 16 respectively. The results are shown in Fig. 10k and l and are consistent with the previous experiments, verifying that even more complex models fail for query points with high RU values.

The correlations between the RU values and the model performance metrics are visually clear in Figs. 9 and 10. Still to further verify these correlations we computed the Pearson correlation between the RU value bins and model performance metrics. The results are provided in Table 1, which confirms the high correlation values for the datasets AD, RN, HS, and DI. Furthermore, to confirm that these values are not sensitive to the binning choices, we perturbed the bin boundaries. However, the results did not meaningfully change, which confirms their robustness.

Summary of Proof of Concept: In short, experiments consistently demonstrate that as the RU values grow, the ML models become less reliable in capturing the truth for the corresponding regions. Consequently, when RU values for a query point in a data set are high, one should discard or at least not rely on the outcome of the model constructed on it for critical decisions.

6.3 Comparison with the existing work

In this section, we thoroughly evaluate the RU measures in the context of the existing approaches discussed in Sect. 7, demonstrate why the existing approaches fail, and how RU measures are superior in capturing the unreliability of individual predictions.

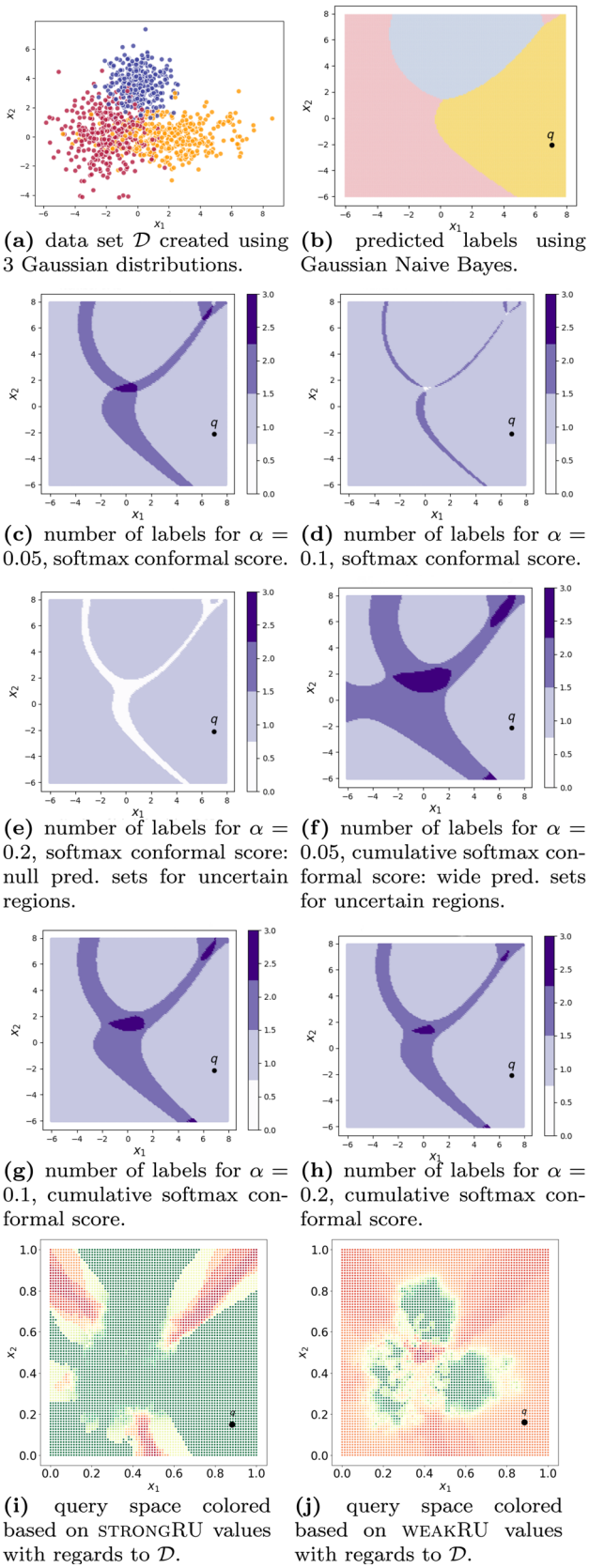
Consider data set $\mathcal{D}$ as shown in Fig. 11a created with three Gaussian distributions representing classes *red, blue, orange*. An arbitrary classification model (e.g. Gaussian Naive Bayes classifier) as the base classifier is trained on $\mathcal{D}$ and the predicted labels are depicted in Fig. 11b. Finally, Fig. 11i and j show the corresponding STRONGRU and WEAKRU values for data set $\mathcal{D}$.

Through the rest of this section, we use the data set $\mathcal{D}$ and the classifier described above to evaluate existing methods for the reliability of individual predictions.

6.3.1 Conformal prediction

We start by employing the conformal prediction (CP) framework[9] with confidence level α of 0.2, 0.1, and 0.05 and softmax score output of the base classifier as the conformity score. Results are shown in Fig. 11c–e. As can be seen in Fig. 11e, CP is creating empty prediction sets for $\alpha = 0.2$ for query points around the uncertain areas which are faulty and show that CP is highly dependent on the choice of α. The null region disappears for smaller α values but ambiguous classification regions arise with several labels included in the prediction sets highlighting the uncertain behavior of the base classifier. By choosing the cumulative softmax conformal score, the empty prediction set problem is resolved however, uncertain regions are emphasized by wider boundaries. Now consider the query point q; according to the model prediction, q belongs to the *orange* class and regardless of the chosen α, CP confirms that. However, this is only true if the true decision boundary is identical to the one estimated by the base classifier [70]; still, as previously discussed in Sect. 3.1, this may not always be the case. Therefore, although q is in an uncertain region, CP fails to capture it as it always returns a prediction set of size 1. Conversely, as can be seen in Fig. 11j, the WEAKRU measure can successfully capture the RU associated with q as it is an outlier, yet does not belong to an uncertain region.

[9] We used MAPIE (Model Agnostic Prediction Interval Estimator), which is an implementation of Conformal Prediction works such as [5, 67]. See more at: https://mapie.readthedocs.io/



(a) data set $\mathcal{D}$ created using 3 Gaussian distributions. **(b)** predicted labels using Gaussian Naive Bayes. **(c)** number of labels for $\alpha = 0.05$, softmax conformal score. **(d)** number of labels for $\alpha = 0.1$, softmax conformal score. **(e)** number of labels for $\alpha = 0.2$, softmax conformal score: null pred. sets for uncertain regions. **(f)** number of labels for $\alpha = 0.05$, cumulative softmax conformal score: wide pred. sets for uncertain regions. **(g)** number of labels for $\alpha = 0.1$, cumulative softmax conformal score. **(h)** number of labels for $\alpha = 0.2$, cumulative softmax conformal score. **(i)** query space colored based on STRONGRU values with regards to $\mathcal{D}$. **(j)** query space colored based on WEAKRU values with regards to $\mathcal{D}$.

Fig. 11 Conformal prediction fails to detect the prediction unreliability for not well-represented point **q** while WEAKRU correctly captures such unreliability

6.3.2 Prediction probabilities

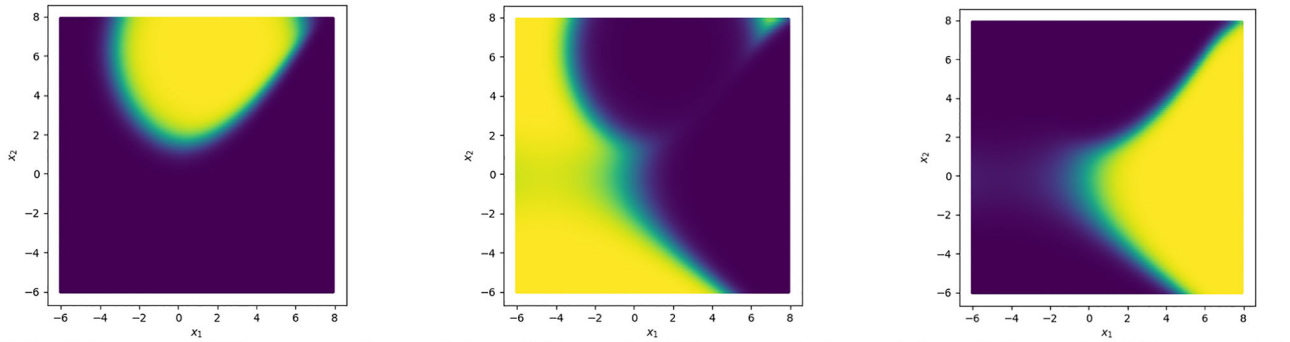
In the next experiment, we evaluate the prediction probabilities generated by probabilistic classification models and demonstrate their failure for query points that are not represented by data. To do so, we employ data set $\mathcal{D}$ and train an arbitrary probabilistic classifier such as Gaussian Naive Bayes on it (remember that we can use any classifier, however, if the model is not intrinsically probabilistic, we need to make sure that the probabilistic outcomes are calibrated). Figure 12a–c show the prediction probabilities assigned to either of the classes *red, blue* and *orange*. As observed, prediction probabilities fail to capture query points that belong to unrepresented regions and assign a negligible chance of belonging to any other class but the one determined by the decision boundary, however, this is only true if the true decision boundary is identical to the one estimated by the base classifier and as previously discussed in Sect. 3.1, this may not always be the case if the distribution between training and production data vary.

6.3.3 Data coverage

Finally, we conduct an experiment to assess the capacity of data coverage techniques to create proper warnings and demonstrate their failure for query points that are in uncertain regions. To this end, using the continuous notion of data coverage [10] and tuned parameters of $k = 50$ and $\rho = 0.08$, we identify the uncovered region on data set $\mathcal{D}$ as illustrated in Fig. 13. The training data points ($\mathcal{D}$) are highlighted as black dots. The regions highlighted in red and green comprise the uncovered and covered regions respectively. Any query point belonging to the green region is covered and all the query points in the red region are uncovered. While the uncovered region can raise warning signals for the unreliability of underrepresented query points, it fails to capture the unreliability associated with the uncertain regions (regions close to the decision boundary in the case of data set $\mathcal{D}$). Furthermore, even for the query points in underrepresented regions, data coverage creates a binary value that is sensitive to the choice of parameters such as the radius ρ. This issue is further highlighted considering the sharp transition from uncovered to covered, specified by the uncovered region's decision boundary. As a result, while two points close to each other, where one is inside and the other outside of the decision boundary, are almost equally miss-presented, for one the output signal is covered (no warning at all) while the other is uncovered (maximum warning).

6.3.4 Uncertainty sampling

Uncertainty sampling is yet another model-centric notion of uncertainty quantification, which is the dominant approach



(a) Prediction probabilities generated by the model for *blue* class.

(b) prediction probabilities generated by the model for *red* class

(c) prediction probabilities generated by the model for *orange* class

Fig. 12 Prediction probabilities of classifiers trained on $\mathcal{D}$ in Fig. 11a fails for query points that are not well-represented

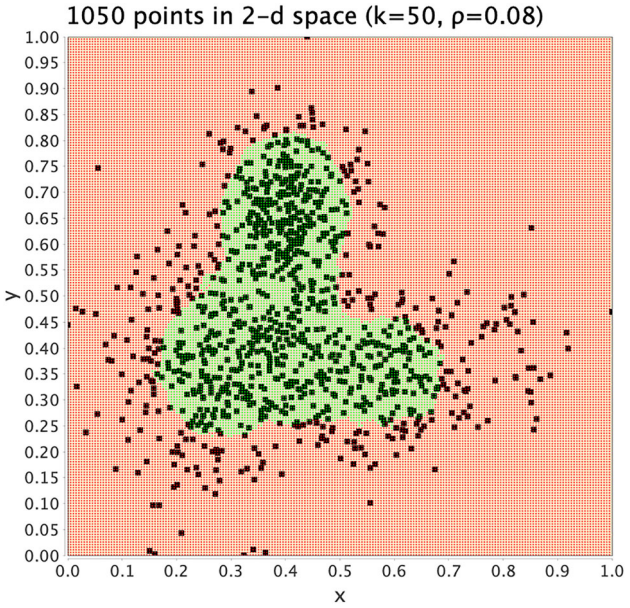


Fig. 13 Data coverage on data set $\mathcal{D}$ in Fig. 11a fails to capture the unreliability associated with the query points in uncertain regions. The training data ($\mathcal{D}$) are highlighted as black dots. The regions highlighted in red and green comprise the uncovered and covered regions respectively. Any query point belonging to the green (red) region is considered (un)covered

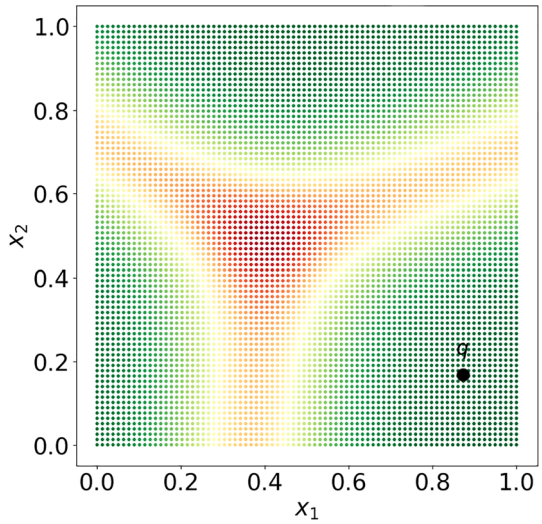


Fig. 14 Query space colored based on model's Shannon entropy values w.r.t. $\mathcal{D}$

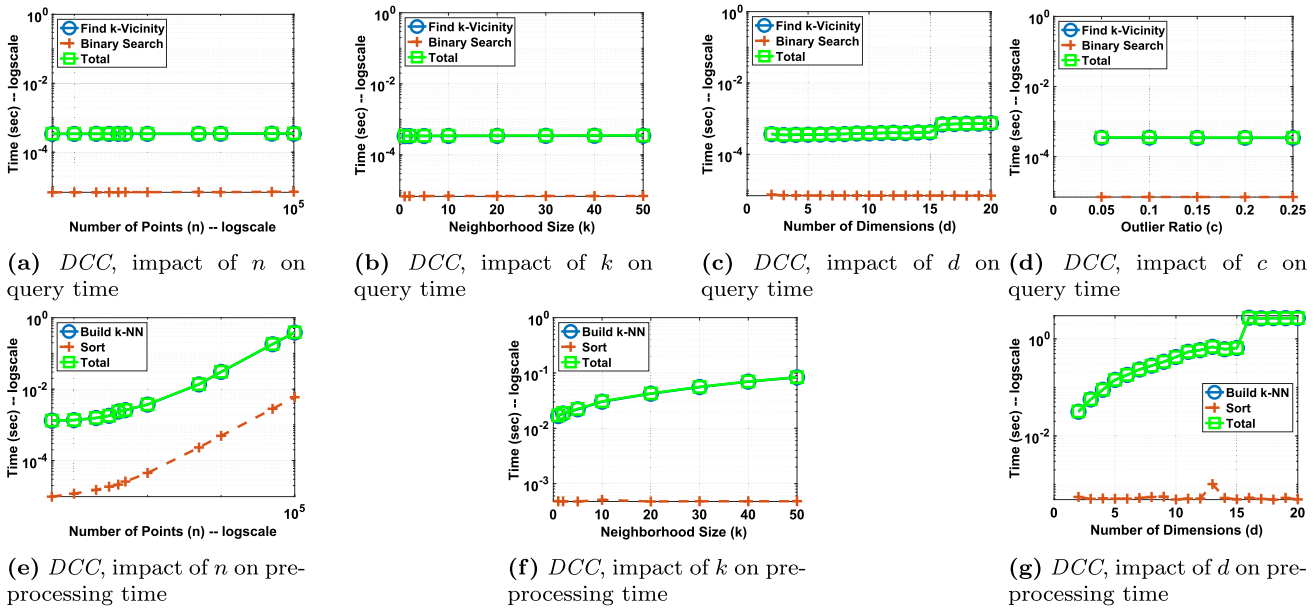
for active learning. In an active learning setting, the model is given a pool of unlabeled samples and should decide which sample to label next. In uncertainty sampling, the idea is to label the sample that the model is most uncertain about. This uncertainty is evaluated based on the model's (probabilistic) confidence for each training point using Shannon entropy. In this experiment, we used the class probabilities assigned by the base classifier to calculate the Shannon entropy for each query point. Figure 14 shows the query space colored based on the model's confidence (using Shannon entropy). As expected, besides being dependent on the model itself, this approach fails for query points that lack sufficient representation.

6.4 Performance evaluations

After demonstrating the effectiveness of the RU measures, we now focus on the performance of our algorithms. In this section, we use the *DCC* and *RN* data sets to evaluate the time efficiency of algorithms. We obtained similar results with almost identical plots for both classification and regression tasks. In the following, we present the results for classification tasks using *DCC* data set with different settings.

6.4.1 Query time

The query time consists of (i) the time to find the k-vicinity of the query point q and identifying the tuple in k-vicinity



(a) *DCC*, impact of n on query time

(b) *DCC*, impact of k on query time

(c) *DCC*, impact of d on query time

(d) *DCC*, impact of c on query time

(e) *DCC*, impact of n on preprocessing time

(f) *DCC*, impact of k on preprocessing time

(g) *DCC*, impact of d on preprocessing time

Fig. 15 Performance evaluation results

that has the maximum distance from q, and (ii) the time to apply binary search on the sorted multi-sets.

Varying :*n* To study the impact of the number of tuples n on the performance of the query time, we gradually increase the size of the data set from 50 to 100K. The results are provided in Fig. 15a. The total query time is dominated by the first bottleneck and the time to binary search the lists is negligible compared to it. In our experiments, the query time did not (meaningfully) change as the data set size increased, showing the scalability of our algorithm to the very large settings.

Varying :*k* Next, we vary the neighborhood size k from 1 to 50. The results in Fig. 15b suggest that the query time is (almost) independent from the k.

Varying :*d* We next study the impact of the number of attributes d by varying it from 2 to 20. The results in Fig. 15c verify the scalability of our algorithms concerning the number of dimensions.

Varying :*c* In our final experiment, we change c from 0.05 to 0.25. The results are brought in Fig. 15d. Results verify that the query time is independent of c.

6.4.2 Preprocessing time

Our preprocessing time consists of two parts. The first is the time to build the k-NN data structure, identifying the k-vicinity radius (in Algorithm 1) and computing uncertainty (in Algorithm 2) for each tuple in the data set. The second one is the time to construct the sorted multi-sets of k-vicinity radii and uncertainty values. We use the exact k-vicinity radii and entropy values in this experiment.

Varying :*n* In this experiment, we study the impact of the number of tuples n in the data set on the preprocessing time by gradually increasing the size of the data set from 50 to 100K. We then measure the time to build the k-NN data structure and construct the sorted multi-sets. The results are provided in Fig. 15e. The total preprocessing time is dominated by the time to build the k-NN data structure and the time to build the multi-sets is almost negligible compared to it. Nevertheless, the cumulative preprocessing time was small enough that the algorithm could scale to larger settings, finishing in less than a second for n=100K.

Varying :*k* To study the impact of neighborhood size k on the preprocessing time, we vary k from 1 to 50. The results can be seen in Fig. 15f. Similarly, the total preprocessing time in this experiment is also dominated by the time to build the k-NN data structure and the algorithm was efficient in all settings, finishing in less than a fraction of a second for k=50.

Varying :*d* To study the impact of the number of attributes d of the data set on the preprocessing time, we gradually change d from 2 to 20. The results are brought in Fig. 15g. Like the previous settings, the total preprocessing time is dominated by the time to build the k-NN data structure and the algorithm linearly scales to larger settings, finishing in less than 3 s for d=20.

6.5 Impact of distance measures on RU values

In this experiment, we study the effect of the distance metric chosen to determine the neighborhood of a data point (in k-NN component) on the RU values. To do so, we employ data set $\mathcal{D}$ (Fig. 9a) and calculate the RU values for the entire query

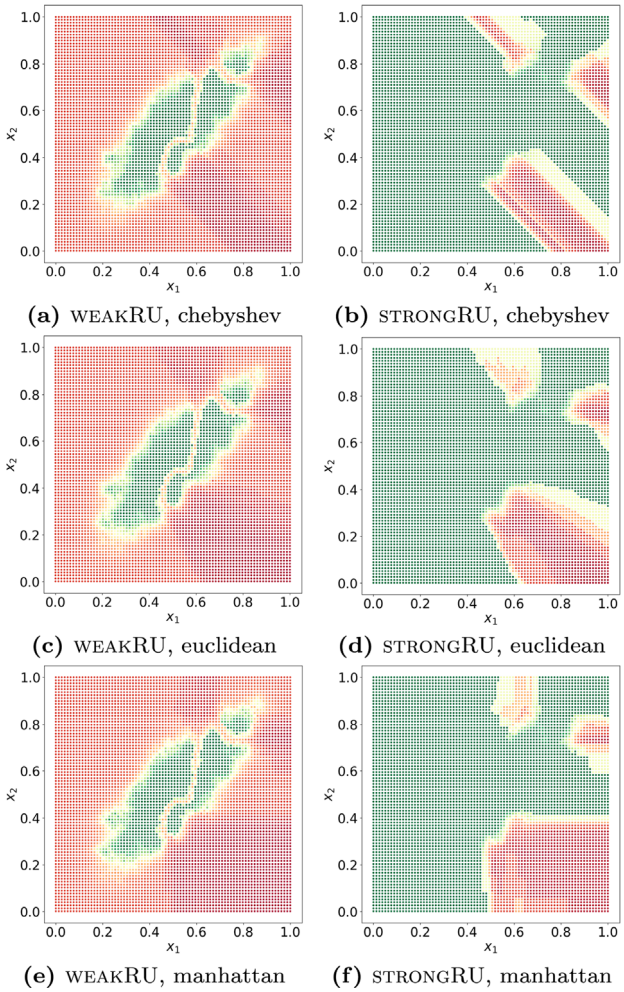


Fig. 16 Query space colored based on WEAKRU and STRONGRU values with regards to $\mathcal{D}$ in Fig. 9a subject to different distance metrics

space w.r.t. 3 distance measures of *chebyshev, manhattan* and *euclidean*. Although the distance metric is expected as an input in our implementation, however, the results show general consistency across both measures (Fig. 16).

6.5.1 Training regression models for no data access

As our final experiment, we study the efficiency of the training process for building the regression models to estimate the values of k-vicinity radius and entropy. As explained in Sect. 4.3, we apply exponential sampling to generate the right amount of data such that the trained models satisfy the user-specified error. As a heuristic, we generate a fraction of samples uniformly and the others from the underlying distribution of the training data using GAN methods or Gaussian copula distribution functions [61]. Figure 17 illustrates the monotonic drop in the error of both regression models trained to predict the entropy and k-vicinity radii of query points (as discussed in Sect. 4.3), as the number of synthetic

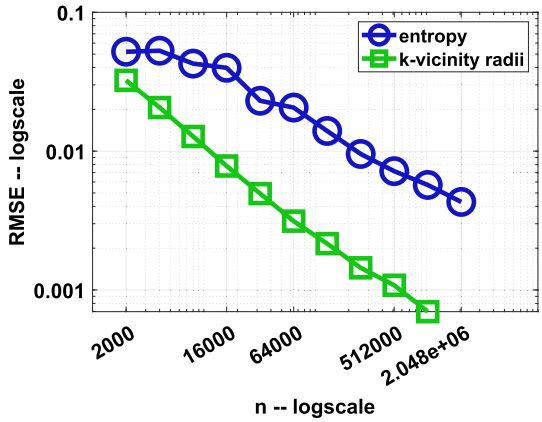


Fig. 17 Effectiveness of exponential search in reducing the error of learning distrust parameters

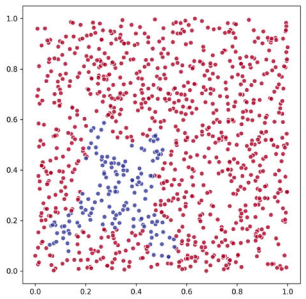
i.i.d samples increases. Since both entropy and k-vicinity radii values are in the range of [0,1], selecting a sufficiently small (RMSE) error threshold (e.g. 10^{-3} or 0.01% average difference between the actual and predicted values) guarantees not going overboard with generating too many samples (affecting preprocessing time) while achieving good prediction accuracy.

6.6 Sensitivity asnalyses

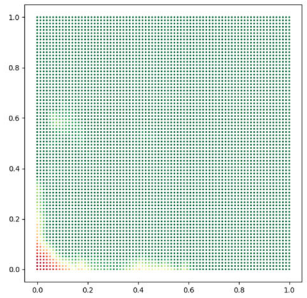
In the following, we provide the complimentary experiments by choosing alternative settings compared to the previous experiments. In short, in the complimentary experiments, we observed consistency with our previous results, further validating and verifying our proposal.

6.6.1 Proof of concept experiments with various underlying distributions

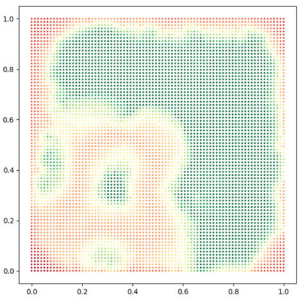
We used Gaussian as the underlying distribution of our synthetic datasets. In this experiment, we study whether the underlying distribution of the data would affect the capacity of the RU measures in revealing unreliability. To do so, we follow the same procedure outlined in the construction of synthetic datasets in Sect. 6.1.2, however, instead of generating a sample following a Gaussian distribution, we opt to utilize alternative distributions such as *Uniform, Exponential, and Logistic*. The results are illustrated in Fig. 18. In summary, aligned with our previous results, there is a consistent correlation between RU values and a model's potential to predict correctly, with models exhibiting more error as RU values increase. For the Uniform distribution (Fig. 18a–e), we expected the entire query space should be equally represented by the training data and hence, the lack of representation scores to be universally low. As a result, the STRONGRU scores remain low. WEAKRU in this case, mostly reflects the



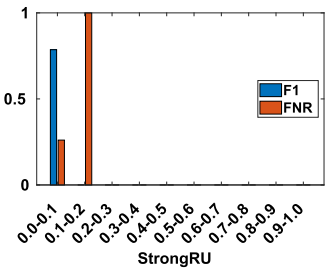
(a) a data set $\mathcal{D}_u$ in SYN following a uniform distribution



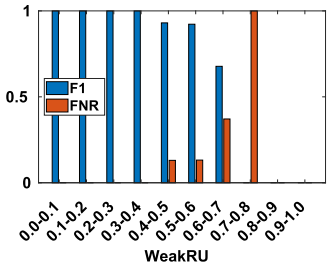
(b) query space colored based on STRONGRU values w.r.t. $\mathcal{D}_u$ in Fig. 18a



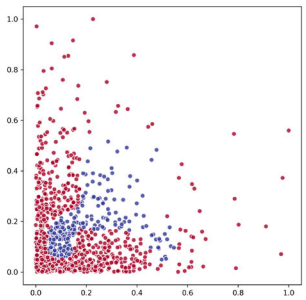
(c) query space colored based on WEAKRU values w.r.t. $\mathcal{D}_u$ in Fig. 18a



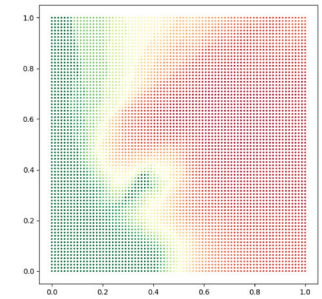
(d) effectiveness of STRONGRU over $\mathcal{D}_u$ in Fig. 18a



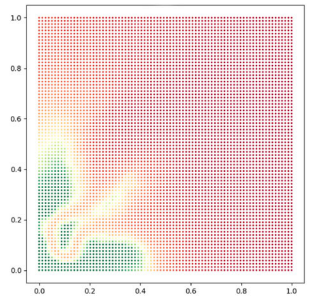
(e) effectiveness of WEAKRU over $\mathcal{D}_u$ in Fig. 18a



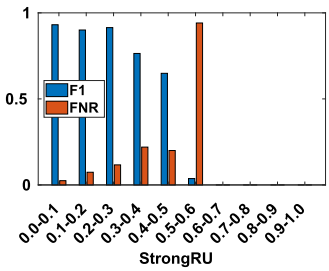
(f) a data set $\mathcal{D}_e$ in SYN following a exponential distribution



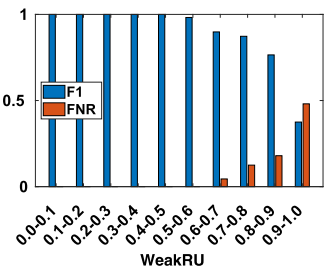
(g) query space colored based on STRONGRU values w.r.t. $\mathcal{D}_e$ in Fig. 18f



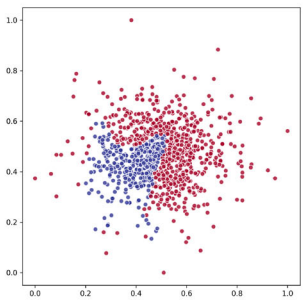
(h) query space colored based on WEAKRU values w.r.t. $\mathcal{D}_e$ in Fig. 18f



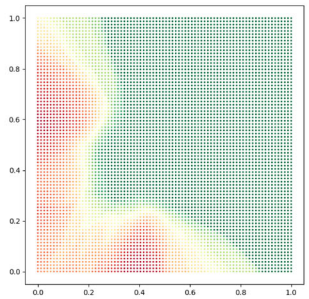
(i) effectiveness of STRONGRU over $\mathcal{D}_e$ in Fig. 18f



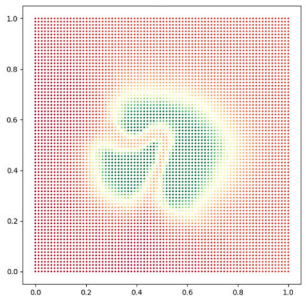
(j) effectiveness of WEAKRU over $\mathcal{D}_e$ in Fig. 18f



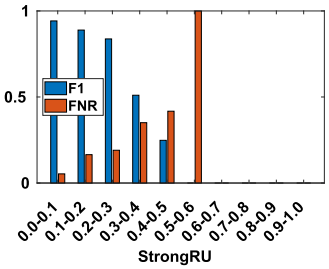
(k) a data set $\mathcal{D}_l$ in SYN following a logistic distribution



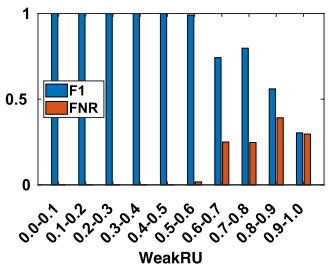
(l) query space colored based on STRONGRU values w.r.t. $\mathcal{D}_l$ in Fig. 18k



(m) query space colored based on WEAKRU values w.r.t. $\mathcal{D}_l$ in Fig. 18k



(n) effectiveness of STRONGRU over $\mathcal{D}_l$ in Fig. 18k



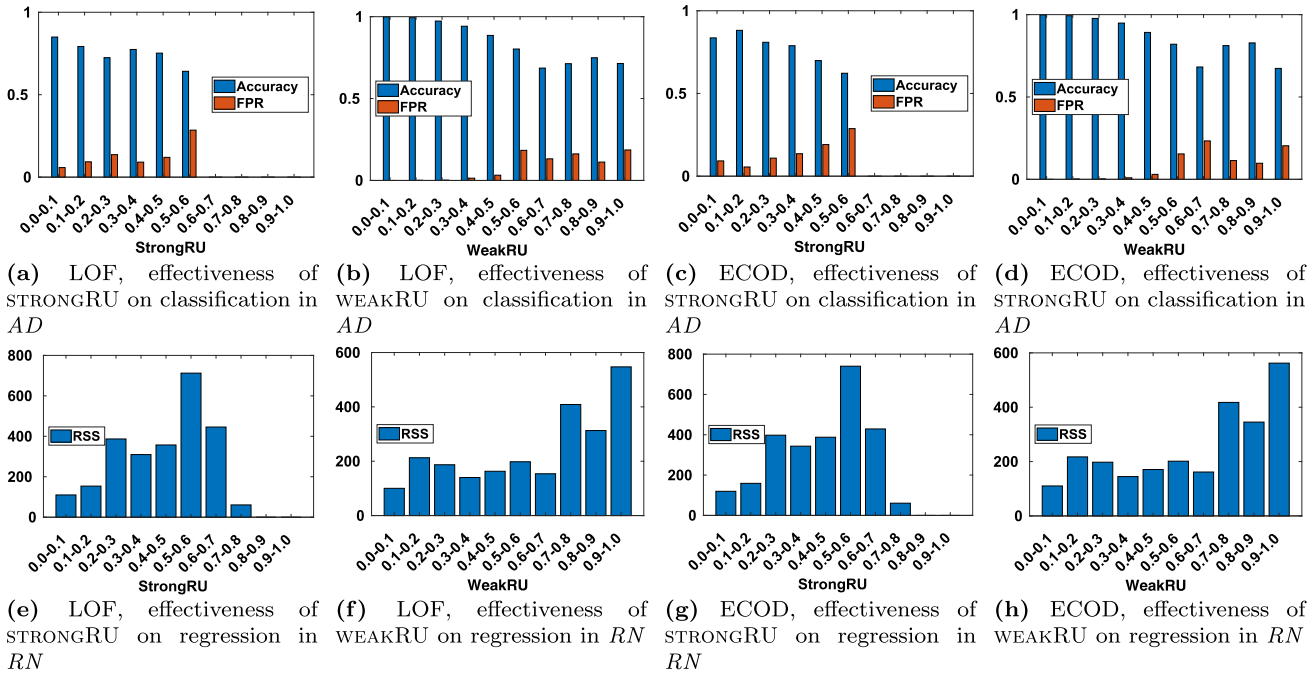
(o) effectiveness of WEAKRU over $\mathcal{D}_l$ in Fig. 18k

Fig. 18 effect of sampling following different distributions while generating SYN data sets on distrust values

impact of uncertainty. Consistent with our previous results, one can see the high correlation between WEAKRU and model performance in Fig. 18e.

6.6.2 Effect of outlier detection metric

As previously noted in Sect. 4, the RU measures are agnostic to the choice of the outlier detection technique. While we



(a) LOF, effectiveness of STRONGRU on classification in *AD*
(b) LOF, effectiveness of WEAKRU on classification in *AD*
(c) ECOD, effectiveness of STRONGRU on classification in *AD*
(d) ECOD, effectiveness of STRONGRU on classification in *AD*
(e) LOF, effectiveness of STRONGRU on regression in *RN*
(f) LOF, effectiveness of WEAKRU on regression in *RN*
(g) ECOD, effectiveness of STRONGRU on regression in *RN*
(h) ECOD, effectiveness of WEAKRU on regression in *RN*

Fig. 19 Effect of varying the lack of representation component method of choice on the distrust values

used the distance of $k-$th nearest tuple to estimate $\mathbb{P}_o$, in this experiment, we investigate the impact of using alternative outlier detection approaches in computing RU values. More specifically, we integrate the following two outlier detection metrics in the lack of representation oracle:

- Local Outlier Factor (LOF) [16]: Widely used proximity based outlier detection technique.
- Empirical Cumulative Distribution-based Outlier Detection (ECOD) [48]: The SOTA probabilistic outlier detection technique recognized by the PyOD toolbox [87].

We conduct this experiment on one classification (using *AD* data set) and one regression task (using *RN* data set). The results are shown in Fig. 19. In short, by comparing the results corresponding to the same task and data set (e.g. effectiveness of STRONGRU results for *AD* as shown in Fig. 19a and c and previously in 10a and c), we observe almost identical results per RU value ranges. This pattern is repeated for the effectiveness of WEAKRU results for *AD* (Fig. 19b and d and previously in 10b and d) and the effectiveness of STRONGRU and WEAKRU results for *RN* (Fig. 19e–h and previously Fig. 9l and m), demonstrating the flexibility of RU measures to the choice of the outlier detection technique when computing $\mathbb{P}_o$.

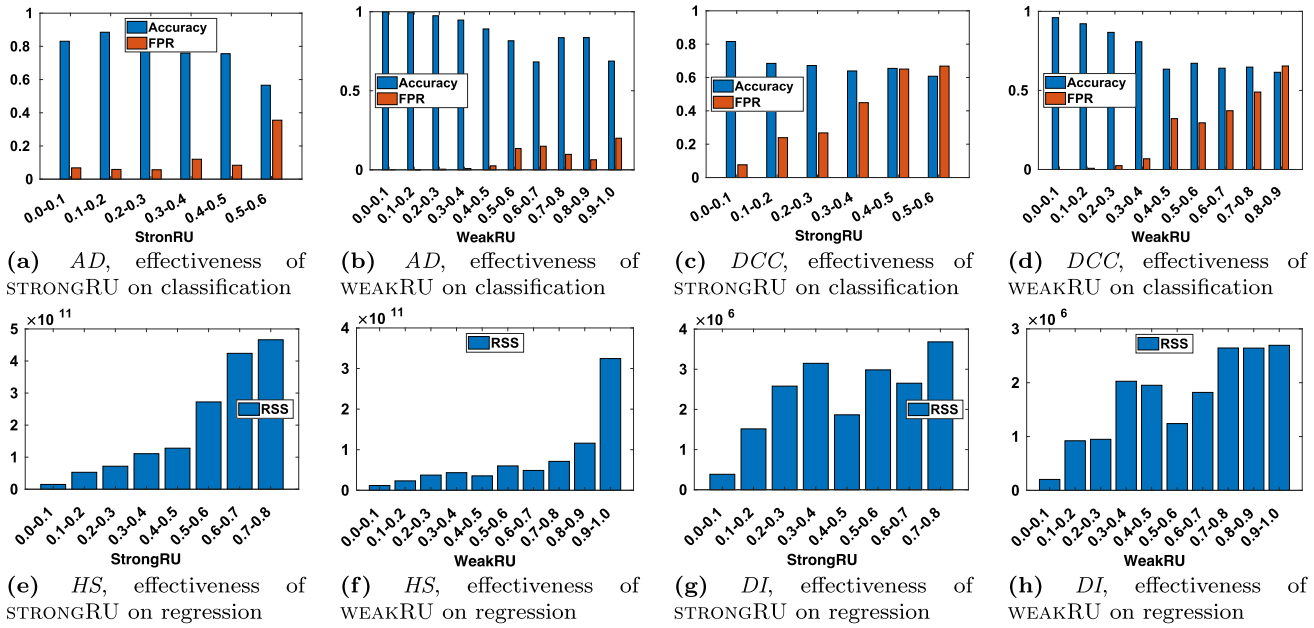
6.6.3 Proof of concept experiments with uniformity of train/test samples

In Sect. 6.1.2, to overcome the challenge of collecting enough samples to evaluate the effectiveness of RU measures, we proposed sub-sampling from a large data set, removing the outliers and adding the outliers back to the test set to cover larger parts of the query space. However, as mentioned in Sect. 6.1.2, the downside of this approach is that it further reduces the presentation of points from under-represented regions in the training set, which may impact the model performance for those regions. Alternatively, in this experiment, we uniformly sample two sets from the underlying distribution. The first set serves as the training set and we did not remove the outliers from it. The second set is used for creating the test set by finding its outliers and adding to the test set. We repeat this experiment for two classifications (*AD* and *DCC*) and two regression (*HS* and *DI*) data sets. The results are illustrated in Fig. 20. In summary, the results are consistent with the previous observations, confirming the validity of our previous experiment and the minimal impact of removing the outliers from the training set to the results.

7 Related work

Responsible data science has become a timely topic, to which the data management community has extensively contributed [8, 43, 68, 69].

In particular, [28] introduces a data profiling primitive *conformance constraint* to characterize whether inference over a tuple is untrustworthy. By the assumption that the *conformance constraints* always hold, they claim that they can use a tuple's deviation from the constraint as a proxy to trust a model's outcome for that tuple. Besides, extensive studies



(a) *AD*, effectiveness of STRONGRU on classification (b) *AD*, effectiveness of WEAKRU on classification (c) *DCC*, effectiveness of STRONGRU on classification (d) *DCC*, effectiveness of WEAKRU on classification

(e) *HS*, effectiveness of STRONGRU on regression (f) *HS*, effectiveness of WEAKRU on regression (g) *DI*, effectiveness of STRONGRU on regression (h) *DI*, effectiveness of WEAKRU on regression

Fig. 20 Uniformly sampled training data and outliers removed from test set

line of work	target	fidelity	output	task	components	model advocacy	data profiler
RU measures (this work)	data	local	probabilistic score	classification regression	lack of certainty lack of representation	challenge	yes
prediction probabilities [84, 85, 64, 59]	model	global	probalistic score	classification	lack of certainty	challenge	no
prediction intervals [40, 62, 21]	model	global	interval	regression	lack of certainty	challenge	no
conformal prediction [7, 70]	model	global	interval set	classification regression	lack of certainty	challenge	no
uncertainty sampling [47, 72]	model	global	continuous score	classification regression	lack of certainty	challenge	no
data coverage [9, 10, 49]	data	local	binary signal	classification regression	lack of representation	challenge	yes
local interpretation [52, 66, 23]	model	local	prediction probability feature effect	classification regression	n/a	support	no
out-of-distribution generalizability [19]	model	global	continuous score	classification regression	lack of representation	challenge	yes

Fig. 21 Descriptive comparison of RU measures and related work

on different dimensions of trust in ML and AI have been presented in [39, 51]. It is also worth mentioning the body of work on the notion of trustworthiness of data sources that focuses on the correctness and legitimacy of data sources[24, 35], however despite the similar terminology, it is a different concept from our problem.

Related work also includes [1, 12, 60, 86] that aim to estimate and quantify uncertainty in AI models, however, they have a different perspective on the issue as they extract the uncertainty from models, while our measures are data-centric. Probabilistic classifiers predict a probability distribution over the set of classes for a given query point instead of simply returning the most likely class [59, 64, 84, 85]. A given probability metric such as log loss or Brier score is calculated for each example to evaluate the predicted probabilities. Not all of the common classifiers are intrinsically probabilistic and some return distorted probabilities that need to be calibrated. Prediction probabilities are computed using the model trained for global performance and may not be accurate for the unrepresented regions. Prediction Intervals (PIs) are a common practice for quantifying the uncertainty associated with a model's prediction of a query point in regression tasks [21, 40, 62]. PIs consist of a lower and upper bound that contains a future observation with a specified level of confidence. Although PIs can be constructed in multiple ways, there is a negative correlation between the quality of the PI and the computational load associated with it [41]. Conformal Prediction (CP) is another standard way of quantifying uncertainty in both classification and regression problems returning confidence intervals and confidence sets respectively, guaranteeing a user-specified confidence level. Benefiting from a heuristic notion of uncertainty in the model of choice, a scoring function $s(x, y)$ is defined that assigns uncertainty values to query point x given target

variable y with larger values to the cases that x and y disagree more. Next, the $1-\alpha$ quantile (α being the user-specified confidence level) of the calibration set scores is calculated and used to form the prediction set for the new examples. Uncertainty sampling [47, 72] is also a related model-centric strategy used in active learning to select the most informative data points for annotation or labeling. The goal of active learning is to reduce the labeling cost by selecting the most valuable data instances to be labeled, rather than labeling all available data. Uncertainty sampling achieves this by selecting data points about which the model is uncertain or has low confidence in its predictions. The model typically measures uncertainty through metrics like entropy, margin, or least confidence using the probabilities assigned to different classes.

It is important to note that all aforementioned model-centric approaches, including PI and CP, estimate intervals, probabilities, and scores using model(s) built by maximizing the *expected performance* on *random* sample from the underlying distribution. As a result, while they may provide accurate estimations for the dense regions of data (e.g. majority groups), their estimation accuracy is questionable for the poorly represented regions (e.g. minority groups). In particular, [7] recognizes the lack of guarantees in the performance of CP for such regions. On the contrary, prediction outcomes are specifically unreliable for regions that are unlikely to be sampled. As a result, as we further discuss in Sect. 3.1, such approaches fail for cases that are not represented by the training data. This is consistent with our experimental evaluations. PI and CP methods usually rely on techniques such as bootstrapping and constructing ensembles to elicit uncertainty, which regardless of the number of subsamples or ensembles created, fails to account for the regions that are not represented. Contrarily, our proposed measures are computed *locally* around the query point (in form of lack of certainty and lack of representation) and therefore are equally accurate for different regions of data. Finally, while PI and CP return an interval or set for each query point, the results may be too generic (e.g. including a large set) or lack a proper explanation for the user to make an informed decision.

The notion of data *coverage* is a related topic that has been studied across different settings [2, 3, 9, 10, 36, 49, 57, 78]. For categorical data, uncovered regions are identified in form of value combinations (e.g. Hispanic Females) called patterns. A pattern is uncovered if there are not enough samples matching it [9, 36, 49]. *Coverage* on continuous space is studied in [10]. Accordingly, lack of *coverage* is identified as any point in the data space that doesn't have enough points in a fixed-radius neighborhood around it. Although coverage does not provide a score for an arbitrary query point, following the idea of whether the point is covered or not, users can decide whether to trust the outcome of the model for that query point.

Out-of-distribution generalizability is another related topic from the ML community that quantifies the degree to which a query point is an outlier in the underlying distribution. Specifically, [19] proposes five metrics for identifying well-represented examples. These metrics are shown to be highly correlated, stable, and model-agnostic. The metrics rank examples based on different measures within ensembles, distance to the decision boundary, or prediction difference of two models for the same query point (holdout retraining). It is important to note that these techniques are model-agnostic in the sense that they have consistent results for different models and parameters, however, unlike our techniques that merely assess representation from the data, they still measure representation within model properties.

Another related topic is the body of work on local interpretation methods for explaining individual predictions [55]. LIME provides local explanations for a model's prediction behavior on query points by substituting the original complex model with a locally interpretable surrogate model. Being a model-agnostic technique, to realize what parts of the input are involved in the prediction, LIME perturbs the query point by creating samples around its neighborhood and observes how the model performs for the perturbed samples. Next, the samples are weighted with regard to their proximity to the original query point, and an interpretable model is constructed on the new samples. The learned model should be locally a good approximation and be used to interpret the original model. We note that interpretation methods justify a model's reasoning for a particular behavior. Conversely, our measures raise warnings to cast doubt when the prediction outcome is not reliable for a specific case.

SHAP (Shapley Additive Explanations) [52] is another model-agnostic framework for explaining individual predictions made by machine learning models. SHAP values are based on cooperative game theory concepts, specifically the Shapley value, which allocates a fair contribution to each feature in the prediction. SHAP assigns importance values to input features, indicating their contribution to a model's prediction. It also can provide both local explanations for individual predictions and global insights into overall model behavior. Similar to SHAP, QII (Quantitative Input Influence) [23] also uses Shapley values to explain individual predictions, yet, instead of adopting the conditional approach used in SHAP, QII draws ideas from the causal inference and follows an interventional approach. The QII method addresses feature correlations by iteratively altering individual features and calculating the average impact of each change on the model's output, considering all features used in constructing the model.

Figure 21, presents an extensive comparison between the related body of work and our proposed measures and demonstrates how our measures stand out in the skyline. The techniques are examined based on the following properties:

- *Target* specifies whether the technique targets data or model.
- *Fidelity* specifies whether the technique evaluates trust locally or only provides global assurance (may fail for sparse regions in the data).
- *Output* specifies the outcome of the technique.
- *Task* specifies the learning problem.
- *Component* specifies the considered complications causing the trust problems.
- *Model advocacy* specifies whether the technique questions the outcome of the model or tries to justify it.
- *Data profiler* specifies whether or not the outcome of the technique is considered as a property of the data.

To the best of our knowledge, our paper is the first to provide data-centric RU measures to identify the scope of use of data sets for individual predictions. The techniques proposed in this paper rely on the extensive research and advanced algorithms for outlier detection [16, 34, 65, 82] and uncertainty computing [15, 17, 29, 71].

8 Conclusion

Towards addressing the need for trustworthy AI, in this paper, we proposed RU measures as the warning signals that limit datasets' scope of use for predicting future query points. These measures are valuable alongside other techniques for trustworthy AI. We proposed novel ideas for the effective implementation of RU measures and designed efficient algorithms that scale to very large, high-dimensional data. Our comprehensive experiments on real-world and synthetic data sets validated our proposal and verified the scalability of our algorithms with sub-second run times.

Acknowledgements This research was support by NSF 2107290. The authors would like to thank the reviewers and the meta-reviewer for their constructive feedback. We would also like to thank Dr. Saravanan Thirumuruganathan and Prof. H. V. Jagadish for their valuable insights.

References

1. Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U.R., Makarenkov, V., Nahavandi, S.: A review of uncertainty quantification in deep learning: Techniques, applications and challenges. Information Fusion pp. 243–297 (2021)
2. Accinelli, C., Catania, B., Guerrini, G., Minisi, S.: The impact of rewriting on coverage constraint satisfaction. In: EDBT/ICDT Workshops (2021)
3. Accinelli, C., Minisi, S., Catania, B.: Coverage-based rewriting for data preparation. In: EDBT/ICDT Workshops (2020)
4. Agarwal, P.K., De Berg, M., Matousek, J., Schwarzkopf, O.: Constructing levels in arrangements and higher order Voronoi diagrams. SIAM J. Comput. **27**(3), 654–667 (1998)
5. Agarwal, P.K., Kumar, N., Sintos, S., Suri, S.: Efficient algorithms for k-regret minimizing sets. LIPIcs (2017)
6. Agrawal, S.: Diamonds (2017). https://www.kaggle.com/datasets/shivam2503/diamonds
7. Angelopoulos, A.N., Bates, S.: A gentle introduction to conformal prediction and distribution-free uncertainty quantification. CoRR (2021)
8. Asudeh, A., Jagadish, H., Stoyanovich, J., Das, G.: Designing fair ranking schemes. In: SIGMOD, pp. 1259–1276 (2019)
9. Asudeh, A., Jin, Z., Jagadish, H.: Assessing and remedying coverage for a given dataset. In: ICDE, pp. 554–565. IEEE (2019)
10. Asudeh, A., Shahbazi, N., Jin, Z., Jagadish, H.: Identifying insufficient data coverage for ordinal continuous-valued attributes. In: SIGMOD, pp. 129–141 (2021)
11. Blanchard, L., Zhao, B., Yinger, J.: Do lenders discriminate against minority and woman entrepreneurs? J. Urban Econ. **63**(2), 467–497 (2008)
12. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural network. In: ICML, pp. 1613–1622 (2015)
13. Bohler, C., Cheilaris, P., Klein, R., Liu, C.H., Papadopoulou, E., Zavershynskyi, M.: On the complexity of higher order abstract Voronoi diagrams. In: International Colloquium on Automata, Languages, and Programming, pp. 208–219. Springer (2013)
14. Bosnić, Z., Kononenko, I.: Comparison of approaches for estimating reliability of individual regression predictions. Data Knowl. Eng. **67**(3), 504–516 (2008)
15. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Routledge, London (2017)
16. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In: SIGMOD, pp. 93–104 (2000)
17. Brier, G.W., et al.: Verification of forecasts expressed in terms of probability. Mon. Weather Rev. **78**(1), 1–3 (1950)
18. Butler, A.W., Mayer, E.J., Weston, J.: Racial discrimination in the auto loan market. Available at SSRN (2020)
19. Carlini, N., Erlingsson, U., Papernot, N.: Distribution density, tails, and outliers in machine learning: Metrics and applications. CoRR (2019)
20. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. CSUR **41**(3), 1–58 (2009)
21. Chatfield, C.: Prediction intervals. J. Bus. Econ. Stat. **11**, 121–135 (1993)
22. Chazelle, B., Edelsbrunner, H.: An improved algorithm for constructing kth-order Voronoi diagrams. IEEE Trans. Comput. **100**(11), 1349–1354 (1987)
23. Datta, A., Sen, S., Zick, Y.: Algorithmic transparency via quantitative input influence: theory and experiments with learning systems. In: SP, pp. 598–617. IEEE (2016)
24. Dong, X.L., Gabrilovich, E., Murphy, K., Dang, V., Horn, W., Lugaresi, C., Sun, S., Zhang, W.: Knowledge-based trust: estimating the trustworthiness of web sources. CoRR (2015)
25. Dwork, C., Hardt, M., Pitassi, T., Reingold, O., Zemel, R.: Fairness through awareness. In: ITCS, pp. 214–226 (2012)
26. Edelsbrunner, H., Seidel, R.: Voronoi diagrams and arrangements. Discr. Comput. Geom. **1**(1), 25–44 (1986)
27. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, pp. 226–231 (1996)
28. Fariha, A., Tiwari, A., Radhakrishna, A., Gulwani, S., Meliou, A.: Conformance constraint discovery: Measuring trust in data-driven systems. In: SIGMOD, pp. 499-512. Association for Computing Machinery (2021)
29. Gebel, M.: Multivariate calibration of classifier scores into the probability space. Ph.D. thesis, Citeseer (2009)
30. Gunning, D., Aha, D.: Darpa's explainable artificial intelligence (XAI) program. AI Mag. **40**(2), 44–58 (2019)

31. Guyon, I., Gunn, S., Ben-Hur, A., Dror, G.: Result analysis of the NIPS 2003 feature selection challenge. NeurIPS **17**, (2004)
32. Harlfoxem: House sales in king county, USA (2016). https://www.kaggle.com/harlfoxem/housesalesprediction/
33. Harradon, M., Druce, J., Ruttenberg, B.: Causal learning and explanation of deep neural networks via autoencoded activations. CoRR (2018)
34. Hautamaki, V., Karkkainen, I., Franti, P.: Outlier detection using k-nearest neighbour graph. In: Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. (2004)
35. Jayasinghe, U., Otebolaku, A., Um, T.W., Lee, G.M.: Data centric trust evaluation and prediction framework for iot. In: ITU K, pp. 1–7. IEEE (2017)
36. Jin, Z., Xu, M., Sun, C., Asudeh, A., Jagadish, H.: Mithracoverage: A system for investigating population bias for intersectional fairness. In: SIGMOD, pp. 2721–2724 (2020)
37. Kakade, S.M.: On the sample complexity of reinforcement learning. University of London, University College London (United Kingdom) (2003)
38. Kaul, M., Yang, B., Jensen, C.S.: Building accurate 3d spatial networks to enable next generation intelligent transportation systems. In: MDM, vol. 1, pp. 137–146. IEEE (2013)
39. Kentour, M., Lu, J.: Analysis of trustworthiness in machine learning and deep learning. InfoComp (2021)
40. Khosravi, A., Nahavandi, S., Creighton, D., Atiya, A.F.: Lower upper bound estimation method for construction of neural network-based prediction intervals. IEEE Trans. Neural Netw. **22**(3), 337–346 (2010)
41. Khosravi, A., Nahavandi, S., Creighton, D., Atiya, A.F.: Comprehensive review of neural network-based prediction intervals and new advances. IEEE Trans. Neural Netw. **22**(9), 1341–1356 (2011)
42. Kohavi, R., et al.: Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. KDD **96**, 202–207 (1996)
43. Kuhlman, C., Rundensteiner, E.: Rank aggregation algorithms for fair consensus. VLDB **13**(12), 2706–2719 (2020)
44. Kulynych, B., Yang, Y.Y., Yu, Y., Błasiok, J., Nakkiran, P.: What you see is what you get: Distributional generalization for algorithm design in deep learning. CoRR (2022)
45. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010). http://yann.lecun.com/exdb/mnist/
46. Lee, D.T.: On k-nearest neighbor Voronoi diagrams in the plane. IEEE Trans. Comput. **100**(6), 478–487 (1982)
47. Lewis, D.D.: A sequential algorithm for training text classifiers: Corrigendum and additional data. In: SIGIR, pp. 13–19. ACM New York, NY, USA (1995)
48. Li, Z., Zhao, Y., Hu, X., Botta, N., Ionescu, C., Chen, G.: Ecod: Unsupervised outlier detection using empirical cumulative distribution functions. TKDE (2022)
49. Lin, Y., Guan, Y., Asudeh, A., Jagadish, H.: Identifying insufficient data coverage in databases with multiple relations. VLDB **13**(12), 2229–2242 (2020)
50. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: ICDM, pp. 413–422. IEEE (2008)
51. Liu, H., Wang, Y., Fan, W., Liu, X., Li, Y., Jain, S., Jain, A.K., Tang, J.: Trustworthy AI: a computational perspective. CoRR (2021)
52. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. NeurIPS **30**, (2017)
53. McCallum, A.: real-sim data set. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#real-sim
54. McCallum, A.: Sraa data set. https://people.cs.umass.edu/~mccallum/data.html
55. Molnar, C.: Interpretable machine learning. Lulu.com (2020)
56. Montoya, A.M., Parrado, E., Solís, A., Undurraga, R.: Bad taste: gender discrimination in the consumer credit market. Tech. rep., IDB Working Paper Series (2020)
57. Moskovitch, Y., Jagadish, H.: Countata: dataset labeling using pattern counts. VLDB **13**(12), 2829–2832 (2020)
58. Moskovitch, Y., Jagadish, H.: Reliability at multiple stages in a data analysis pipeline. CACM **65**(11), 118–128 (2022)
59. Niculescu-Mizil, A., Caruana, R.: Predicting good probabilities with supervised learning. In: ICML, pp. 625–632 (2005)
60. Pakdaman Naeini, M., Cooper, G., Hauskrecht, M.: Obtaining well calibrated probabilities using bayesian binning. AAAI (2015)
61. Patki, N., Wedge, R., Veeramachaneni, K.: The synthetic data vault. In: DSAA, pp. 399–410 (2016)
62. Pearce, T., Brintrup, A., Zaki, M., Neely, A.: High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In: ICML, pp. 4075–4084. PMLR (2018)
63. Petersen, B., Petersen, T.N., Andersen, P., Nielsen, M., Lundegaard, C.: A generic method for assignment of reliability scores applied to solvent accessibility predictions. BMC Struct. Biol. **9**, 1–10 (2009)
64. Platt, J., et al.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Adv. Large Margin Classif. **10**(3), 61–74 (1999)
65. Ramaswamy, S., Rastogi, R., Shim, K.: Efficient algorithms for mining outliers from large data sets. SIGMOD Rec. pp. 427-438 (2000)
66. Ribeiro, M.T., Singh, S., Guestrin, C.: " Why should i trust you?" Explaining the predictions of any classifier. In: KDD, pp. 1135–1144 (2016)
67. Sadinle, M., Lei, J., Wasserman, L.: Least ambiguous set-valued classifiers with bounded error levels. J. Am. Stat. Assoc. **114**(525), 223–234 (2019)
68. Salimi, B., Howe, B., Suciu, D.: Database repair meets algorithmic fairness. ACM SIGMOD Rec. **49**(1), 34–41 (2020)
69. Salimi, B., Rodriguez, L., Howe, B., Suciu, D.: Interventional fairness: Causal database repair for algorithmic fairness. In: SIGMOD, pp. 793–810 (2019)
70. Shafer, G., Vovk, V.: A tutorial on conformal prediction. JMLR **9**(3), (2008)
71. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**(3), 379–423 (1948)
72. Sharma, M., Bilgic, M.: Most-surely vs. least-surely uncertain. In: ICDM, pp. 667–676. IEEE (2013)
73. Sheikh, M.A., Goel, A.K., Kumar, T.: An approach for prediction of loan approval using machine learning algorithm. In: ICESC, pp. 490–494. IEEE (2020)
74. Sindhwani, V., Keerthi, S.S.: Large scale semi-supervised linear svms. In: SIGIR, pp. 477–484 (2006)
75. Singh, R., Vatsa, M., Ratha, N.: Trustworthy AI. In: IKDD, pp. 449–453. IKDD (2021)
76. Suguna, N., Thanushkodi, K.: An improved k-nearest neighbor classification using genetic algorithm. Int. J. Comput. Sci. Issues **7**(2), 18–21 (2010)
77. Sun, C., Asudeh, A., Jagadish, H., Howe, B., Stoyanovich, J.: Mithralabel: Flexible dataset nutritional labels for responsible data science. In: CIKM, pp. 2893–2896 (2019)
78. Tae, K.H., Whang, S.E.: Slice tuner: A selective data acquisition framework for accurate and fair machine learning models. In: SIGMOD, pp. 1771–1783 (2021)
79. Vapnik, V.N.: An overview of statistical learning theory. IEEE Trans. Neural Netw. **10**(5), 988–999 (1999)
80. Wing, J.M.: Trustworthy AI. CACM **64**(10), 64–71 (2021)
81. Wu, Y., Ianakiev, K., Govindaraju, V.: Improved k-nearest neighbor classification. Pattern Recogn. **35**(10), 2311–2318 (2002)
82. Xu, Z., Kakde, D., Chaudhuri, A.: Automatic hyperparameter tuning method for local outlier factor, with applications to anomaly detection. In: Big Data, pp. 4201–4207. IEEE Computer Society (2019)

83. Yeh, I.C., hui Lien, C.: The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications pp. 2473–2480 (2009)
84. Zadrozny, B., Elkan, C.: Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In: ICML, vol. 1, pp. 609–616. Citeseer (2001)
85. Zadrozny, B., Elkan, C.: Transforming classifier scores into accurate multiclass probability estimates. In: KDD, pp. 694–699 (2002)
86. Zhang, Y., Liao, Q.V., Bellamy, R.K.E.: Effect of confidence and explanation on accuracy and trust calibration in AI-assisted decision making. In: FAccT, pp. 295–305 (2020)
87. Zhao, Y., Nasrullah, Z., Li, Z.: PYOD: a python toolbox for scalable outlier detection. JMLR pp. 1–7 (2019)