# Multi-domain Active Learning for Semi-supervised Anomaly Detection

Vincent Vercruyssen[1](✉), Lorenzo Perini[1], Wannes Meert[1], and Jesse Davis[1]

[1]KU Leuven
firstname.lastname@kuleuven.be

**Abstract.** Active learning aims to ease the burden of collecting large amounts of annotated data by intelligently acquiring labels during the learning process that will be most helpful to learner. Current active learning approaches focus on learning from a *single* dataset. However, a common setting in practice requires simultaneously learning models from *multiple* datasets, where each dataset requires a separate learned model. This paper tackles the less-explored multi-domain active learning setting. We approach this from the perspective of multi-armed bandits and propose the *active learning bandits* (ALBA) method, which uses bandit methods to both explore and exploit the usefulness of querying a label from different datasets in subsequent query rounds. We evaluate our approach on a benchmark of 7 datasets collected from a retail environment, in the context of a real-world use case of detecting anomalous resource usage. ALBA outperforms existing active learning strategies, providing evidence that the standard active learning approaches are less suitable for the multi-domain setting.

**Keywords:** Anomaly detection · active learning · semi-supervised learning · multi-armed bandits

## 1 Introduction

Active learning (AL) attempts to alleviate the time and monetary cost of acquiring labeled data by intelligently deciding exactly which unlabeled instances require a label [20]. One task where active learning can be particularly helpful is in *anomaly detection* (AD), where the goal is to learn a model that can identify anomalous instances in a dataset. While anomaly detection was typically treated an unsupervised learning problem, there is growing evidence that in practice AD algorithms benefit from small amounts of labeled data [25,15]. In particular, labels can help overcome the assumptions encoded in unsupervised AD approaches (e.g., all rare behavior is anomalous) by providing examples of infrequent normal behavior such as maintenance. However, anomalies are rare by nature, making it costly to find and label them. Thus, AL can help select those instances whose label would be most *informative* to the underlying anomaly detector [24].

A drawback to classic AL approaches is that they focus on learning from a single dataset. This contrasts with a scenario that often arises in practice, particularly in anomaly detection, where it is necessary to simultaneously model data

from a fleet of similar yet slightly different entities. As an illustrative example, consider trying to detect anomalous resource usage in a chain of retail stores. Each store is different in terms of its location, size, opening hours, services offered, etc. Thus, each store's resource usage, and the resulting dataset, is characterized by a different marginal distribution [16]. Consequently, what constitutes anomalous behavior is store-dependent, which necessitates a separate detection model per store. Or consider building classifiers to detect the occurrence of blade icing in different wind turbines [27]. Each turbine generates its own data and is different from the other turbines in terms of position, size, etc. Hence, training a single model for use in all stores/turbines would not work. Unfortunately, classic AL strategies are not optimized to deal with multiple datasets simultaneously.

In this paper, we focus on adapting active learning to the multi-domain setting in the context of AD. Given multiple datasets and a global fixed budget for the number of labels that can be acquired across all datasets, our objective is to employ active learning to learn one model for each dataset. The key challenge is to decide how to best divide this budget across the different datasets. Naively spending an equal budget on each dataset is likely to be suboptimal as some datasets will require fewer labeled instances to learn an accurate model than others. Hence, one needs to estimate the *marginal gain* of acquiring another label in each dataset. This is challenging as the marginal gain is diminishing: as more labels are actively acquired in a dataset, each one will have a smaller effect on the learned model's performance. This can be viewed through the prism of the exploration-exploitation trade-off. One needs to spend some labeling effort in each dataset to estimate this marginal gain while simultaneously trying to mostly label the datasets with high gains. We address this challenge from the perspective of *multi-armed bandits* (MAB) and propose the *active learning bandits* (ALBA) method. ALBA maintains an estimate of the marginal gain of querying a label from each dataset over the course of multiple query rounds and queries those labels that optimize the exploration-exploitation trade-off. This yields three differences with the classic AL techniques. First, ALBA can handle multiple datasets. Second, ALBA tracks the marginal gain of acquiring a label from *groups of instances* whereas classic AL tends to estimate the marginal gain of a *single* instance. Third, ALBA computes the marginal gain of an instance's label *after* the oracle has been queried and has provided the instance's true label. To summarize, this paper makes the following contributions:[1]

1. We identify and show how multi-domain active learning and multi-armed bandits are related;
2. We propose an approach to multi-domain active learning that uses *rotting bandits* to cope with the diminishing returns of acquiring labels;
3. We explore theoretically and experimentally how the integration of either a heuristic or a random active learning strategy in ALBA impacts its performance;
4. We empirically demonstrate that ALBA outperforms multiple baselines on 7 real-world datasets about water usage where the task is anomaly detection.

[1] Appendix & Code: `https://github.com/Vincent-Vercruyssen/ALBA-paper`

## 2 Preliminaries

***Multi-domain Dataset*** A *domain* consists of an input space $\mathcal{X}$, a label space $\mathcal{Y}$, and a joint probability distribution over the input-label space pair. By sampling observations from a domain's distribution, we obtain a dataset $D$.

A *multi-domain dataset* $\mathcal{M} = \{D^k\}_{k=1}^K$ consists of $K$ datasets, each sampled from a different underlying domain's distribution. A *multi-domain instance* is denoted as $x_i^k$ and its label as $y_i^k$, i.e., instance $i$ of the dataset $D^k$. We assume the input and label space are the same for each of the $K$ domains.

***Pool-based Active Learning*** Each dataset $D^k \in \mathcal{M}$ contains both labeled and unlabeled instances. In pool-based active learning, one tries to construct a classifier $f^k$ for a dataset $D^k$. Initially, no labels are available. Over the course of subsequent iterations, one unlabeled instance in $D^k$ is chosen to be labeled by the oracle, its label is added to $D^k$, and the classifier is retrained [20].

***Multi-armed Bandits*** The origin of the MAB problem stems from clinical trials [23]. An MAB algorithm is typically given a fixed set of actions $\mathcal{A} = \{1, ..., K\}$ (the arms) and a fixed number of rounds to play $T$ (the budget). In each round $t$, the algorithm has to choose one action $j \in \mathcal{A}$ and receives a single random payoff $r_j$ from the corresponding unknown payoff distribution [2]. When action $j$ is taken for the $n^{\text{th}}$ time, the mean of the payoff distribution is $\mu_j(n)$.

In our active learning setting, the payoff distribution is non-stationary and the expected payoff of an action decreases over time. Thus, for all actions, $\mu_j(n)$ is assumed to be positive and non-increasing in $n$. This corresponds to the *rotting bandit* setting [12]. Let $N_j(t)$ be the number of times action $j$ is taken at round $t$, let $\pi$ be a policy (i.e., an infinite sequence of actions), and let $\pi(t)$ denote the action chosen by policy $\pi$ in round $t$. Then, the goal of the MAB algorithm is to maximize the expected sum of payoffs after round $T$ which equals $\mathbb{E}\left[\sum_{t=1}^{T} \mu_{\pi(t)}\left(N_{\pi(t)}(t)\right)\right]$. MAB algorithms embody the *exploration-exploitation* trade-off, exploiting the best action while spending some time exploring the payoff of each action [2]. In this paper, we use the recent *sliding-window average* (SWA) algorithm as a solver for the non-parametric rotting bandit setting, which comes with strong performance guarantees [12].

## 3 Multi-domain Active Learning

The multi-domain active learning (MDAL) problem for anomaly detection is:

**Given:** A multi-domain dataset $\mathcal{M}$ consisting of $K$ unlabeled datasets, a fixed label budget $T$, and an oracle $\mathcal{O}$ that can provide one label at a time;

**Do:** maximize the performance of each domain's anomaly detector $f^k$ by querying additional labels to the oracle.

The two key challenges are (1) figuring out the marginal benefit of acquiring labels in each dataset of $\mathcal{M}$ (= *labeling payoff*), and (2) dealing with the diminishing returns of labeling additional instances. Some datasets in $\mathcal{M}$ will require less labels to learn an accurate detector than other datasets, i.e., their labeling payoff is higher, but we do not know beforehand which ones.

We propose the *active learning bandits* (ALBA) approach which leverages an MAB algorithm to solve the trade-off between exploration (figuring out which datasets have a high labeling payoff) and exploitation (focusing our labeling efforts on the high-payoff datasets). First, ALBA defines several groups of instances for which to track the labeling payoff (Section 3.1). Second, ALBA updates its estimate of the average labeling payoff of each group using a reward function that measures the impact of labeling an instance from that group on the corresponding anomaly detector (Section 3.2). Third, in each query round ALBA picks a group from which to query an instance using the SWA *rotting bandit* algorithm which can handle non-stationary rewards (Section 3.3). Finally, after choosing a group, ALBA still needs to decide which individual instance of that group to query (Section 3.4). ALBA maintains one detector $f^k$ per dataset in $\mathcal{M}$ that is retrained upon receiving a new label from oracle $\mathcal{O}$. We assume that the cost of querying and labeling is the same and constant for all instances in $\mathcal{M}$, and that the oracle is queried one instance at a time. See Section 3.5 for the algorithm's pseudocode.

### 3.1 Choosing an Action Set

We define a group of instances $G^j$ such that $\forall j$ there exists a $k$ such that $G^j \subset D^k$ and $\forall i \neq j : G^j \cap G^i = \varnothing$. Then, we define the action set $\mathcal{A}$ such that each action $j \in \mathcal{A}$ corresponds to choosing a particular group of instances $G^j$ from which one instance will be queried. The most straightforward idea is to let each of the $K$ datasets in $\mathcal{M}$ be its own group such that $G^j = D^k$. Thus, $|\mathcal{A}| = K$.

However, the distribution of informative instances likely varies substantially within each dataset. Therefore, we propose to first divide each dataset into smaller groups using a clustering algorithm, obtaining a set of $C$ clusters for each dataset. Each action now corresponds to choosing a particular cluster and $|\mathcal{A}| = K \times C$. Note that $\forall j : G^j \subset D^k$. This approach gives the MAB algorithm (Section 3.3) more fine-grained control in selecting different groups of instances and learning their labeling payoff. However, it comes at the cost of increased exploration because the algorithm now has to figure out the reward structure for a larger set of actions.

### 3.2 MAB Reward Function

ALBA keeps track of the labeling payoff for each group of instances $G^j$. After choosing action $j$, and receiving the label for the single selected instance $x_i^j \in G^j$ (Section 3.4), the challenge is to design a reward function that reflects the updated payoff of querying the label for one of the remaining unlabeled instances from this group. We consider two possibilities.

*Entropy of the Predictions.* One measure of the labeling payoff of instance $x_i^j$ is its ability to decrease the overall prediction uncertainty of anomaly detector $f^k$ trained on the dataset to which $x_i^j$ belongs. Thus, the payoff $r_j$ of action $j$ that results in querying $x_i^j$ is computed as the decrease in prediction entropy of the detector after retraining:

$$r_j = \sum_{x \in D^k} \left[ H_{f_+^k}(x) - H_{f^k}(x) \right] \tag{1}$$

where $f_+^k$ represents the detector after retraining with the label of $x_i^j$ provided by the oracle. $H_f(x)$ is the Shannon entropy of the predicted label probability (by detector $f$) that $x$ belongs to one of the classes in $\mathcal{Y}$.

*Cosine Similarity of the Predictions.* A more direct measurement of the labeling payoff of $x_i^j$ looks at how many instances the anomaly detector changes its predicted label for after retraining. If this number is large, the model changed a lot and we can say that labeling $x_i^j$ had a large impact. The payoff $r_j$ of action $j$ that results in querying $x_i^j$ is computed as the cosine similarity between the predicted-label vectors:

$$r_j = 1 - \frac{Y_{f_+^k} \cdot Y_{f^k}}{\|Y_{f_+^k}\| \|Y_{f^k}\|} \tag{2}$$

where $Y_f$ is the vector of predicted labels for a dataset by detector $f$ and contains all 0's or 1's (in our experiments, 0 signifies "normal" and 1 "anomalous"). $f_+^k$ and $f^k$ represent the anomaly detectors trained respectively with and without the queried label.

### 3.3 MAB Algorithm

The MAB algorithm chooses an action $j$ from $\mathcal{A}$ in each query round $t$. In our setting, the number of query rounds is fixed and equal to the label budget $T$, the set of possible actions is fixed, only the payoff of the chosen action is observed at each round ($r_j = 0$ if $j$ is not chosen), and the observed payoffs are bounded to the interval $[0, 1]$. The labeling payoff of a group decreases as more instances from that group are queried and labeled. Intuitively, if most of the instances in a group are labeled, acquiring yet another label will have little effect on the anomaly detector, so the labeling payoff will be close to zero. In contrast, if few or no instances are labeled, observing even one label might greatly improve the detector. Hence, given the non-stationary rewards, ALBA uses the SWA rotting bandit algorithm [12] to choose between different actions in each query round $t$. During the AL loop, SWA tracks the decreasing labeling payoff of each action by estimating a sliding-window average of the obtained rewards with window size $W$, i.e., $\bar{\mu}_j(N_j(t)) = \frac{1}{W} \sum_{n=N_j(t)-W}^{N_j(t)} r_j(n)$ where $r_j(n)$ is the reward obtained when action $j$ is chosen for the $n^{\text{th}}$ time. Initially, this estimate is 0 as ALBA only obtains information about an action's reward distribution *after* querying instances.

### 3.4 Query Selection Strategy

We can query one instance per query round to the oracle. Although the MAB algorithm tells us from which group $G^j$ we should query, it does not inform us which particular instance $x_i^j \in G^j$ to query. The solution is to select query instance $x_i^j$ either *randomly* from $G^j$ (RAND) or *heuristically*, using uncertainty sampling (UC) or another suitable AL method.

Using a random selection strategy results in a better regret bound than using a heuristic strategy, because random payoffs produce a less biased estimate of the average payoff $\mu_j(N_j(t))$ for any action $j \in \mathcal{A}$ at any round $t \leq T$. Let us first assume that the rate of change of the true labeling payoff of an action is not affected by which instance from the corresponding group is queried and labeled in any given round $t$.[2] Then, in a theoretical scenario with infinite instances and infinite budget, randomly collecting labels from each group, computing the rewards, and estimating the average labeling payoff with a sliding window average will result in gradually more accurate estimates of each group's *true* average labeling payoff: $|\bar{\mu}_j(N_j(t)) - \mu_j(N_j(t))| \to 0$ for $t \to \infty$, $j = 1, \ldots, K \times C$. This means that, for a given tolerance error $\varepsilon_j > 0$, there exists a certain necessary cost $c_j \in \mathbb{N}$ such that it is guaranteed that the estimate error is smaller than the tolerance: $|\bar{\mu}_j(N_j(t)) - \mu_j(N_j(t))| < \varepsilon_j$, for all $t \geq c_j$. By taking the total cost $c = \sum_{j=1}^{K \times C} c_j$, and the minimum tolerance $\varepsilon = \min\{\varepsilon_j \colon j \leq K \times C\}$, we can claim that all estimates of the average payoffs are accurate enough, independently of which group is considered: $|\bar{\mu}_j(N_j(t)) - \mu_j(N_j(t))| < \varepsilon$ for all $t \geq c$ and $j = 1, \ldots, K \times C$. After paying cost $c$ (i.e., some number of query rounds), the MAB algorithm begins to pick the optimal actions (exploitation). In contrast, as long as $t < c$, it will sometimes pick sub-optimal actions (exploration). Hence, the lower $c$ (i.e., the faster the estimate of all actions' payoff converges), the lower the expected average regret.

The previous statement is true when any unbiased estimator of the average payoff is used. We now show that estimating the average payoff by *heuristically* selecting the instances, e.g., using the UC selection strategy, results in a biased estimate of the average payoff. Intuitively, this is because in each round $t$ the heuristic strategy picks the instance that yields the largest (potential) reward, resulting in an overly optimistic estimate of the true labeling payoff of each group, forcing the MAB algorithm to spend more time exploring ($c$ is larger).

In the following paragraphs, we fix the index $j$ that refers to a specific action and denote with $\mathcal{W}$ the maximum budget that can be spent on the group $G^j$, corresponding to the number of available payoffs. Additionally, $n$ denotes the number of rounds spent on the given group $G^j$, i.e., for any $n$ there exists a round $t$ such that $n = N_j(t)$. For example, $\mu_j(N_j(t))$ would become $\mu(n)$.

**Proposition 1.** *Let $R_1^n, \ldots, R_{\mathcal{W}-n}^n$ be i.i.d. random variables that take the payoffs as values, such that $\mathbb{E}[R_q^n] = \mu(n)$, for $q = 1, \ldots, \mathcal{W} - n$, $n \in W + 1, \ldots, \mathcal{W}$.*

[2] Because AL typically operates with a small budget and large datasets, this assumption is reasonable as the marginal benefit of labeling each additional instance is small.

*Assume that $\mu(n)$ is a positive non-decreasing function and that $\mu(n)-\mu(n+1)$ does not depend on which instance is queried at round $n$. Then,*

$$\mu(n) \leq \mathbb{E}[\bar{\mu}_{rand}] < \mathbb{E}[\bar{\mu}_{heur}].$$

*Proof.* Without loss of generality, let us assume that the random selection strategy picks the random variable with index $p$. For any $n > W$,

$$\mathbb{E}[\bar{\mu}_{rand}] = \mathbb{E}\left[\frac{1}{W}\sum_{i=n-W}^{n} R_p^i\right] = \frac{1}{W}\sum_{i=n-W}^{n}\mu(i) \geq \mu(n),$$

where the last inequality is due to $\mu(n)$ being non-decreasing. This proves the first inequality.

The heuristic selection strategy always queries the instance that yields the maximum of the available payoffs. Now, $R_*^n = \max(R_1^n, \ldots, R_{\mathcal{W}-n}^n)$ is the random variable getting the maximum payoff at each round $n \in W+1, \ldots, \mathcal{W}$. Such a random variable has a different distribution with respect to $R_1^n, \ldots, R_{\mathcal{W}-n}^n$. With $F$ being the cumulative density function of the payoff random variables, for any value $z \in [0,1]$,

$$\mathbb{P}\left(R_*^n \leq z\right) = \prod_{q=1}^{\mathcal{W}-n} \mathbb{P}(R_q^n \leq z) \Rightarrow F_{R_*^n}(z) = \left(F(z)\right)^{\mathcal{W}-n},$$

which means that the cdf of the maximum payoff is not the same as the cdf of any payoff. Given that $F(z) \leq 1$, with $F$ non-constantly equal to 1, and that it is a non-decreasing function, there exists a minimum value $\hat{z} \in [0,1]$ for which the value of the cdf $F$ equals 1. At the same time, for all $0 \leq z < \hat{z}$, $F(z) < 1$. Because the power of positive values lower than 1 returns smaller values, $F_{R_*^n}(z) = F(z)^{\mathcal{W}-n} < F(z)$ for all $z < \hat{z}$, i.e. $1 - F_{R_*^n}(z) > 1 - F(z)$. Finally we can apply Cavalieri's principle to derive the expected value from the cdf,

$$\mathbb{E}[\bar{\mu}_{heur}] = \int_0^1 \left(1 - F_{R_*^n}\right)\, dz > \int_0^1 (1-F)\, dx = \mathbb{E}[\bar{\mu}_{rand}]$$

which proves the second inequality.

The previous proposition states that taking the maximum rewards in a decreasing fashion results in a biased estimate of the average payoff that is strictly greater than the random selection estimate. Therefore, given $\varepsilon$ and $c$ such that $|\bar{\mu}_{rand}(t) - \mu(t)| < \varepsilon$ for all $t \geq c$, the estimate obtained by $\bar{\mu}_{heur}$ has not accurately estimated $\mu(t)$ yet.[3]

[3] Section 5 provides empirical evidence that the random selection strategy indeed leads to better results than the heuristic strategy. Note that the proof relies on the heuristic strategy being able to rank the instances correctly according to their informativeness. In reality, this ranking is approximate.

**Algorithm 1** ALBA: Active Learning Bandits

1: **Input:** Multi-domain dataset $\mathcal{M}$, label budget $T$, oracle $\mathcal{O}$, number of clusters $C$
2: **Output:** Set of trained anomaly detectors $\mathcal{F}$
3: $\mathcal{A} = \varnothing, \mathcal{F} = \varnothing, t = 0$
4: **for** $D^k \in \mathcal{M}$ **do**
5: $\quad \mathcal{A} = \mathcal{A} \cup \textsc{ClusterData}(D^k, C)$
6: $\quad \mathcal{F} = \mathcal{F} \cup \textsc{TrainDetector}(D^k)$
7: **end for**
8: $\overrightarrow{r} = [0]^{j \in \mathcal{A}}$ ▷ initialize the payoff vector
9: **while** $t < T$ **do**
10: $\quad j = \textsc{Swa}(\mathcal{A}, \overrightarrow{r}, t)$ ▷ choose an action
11: $\quad x_i^j = \textsc{rand}(G^j)$ ▷ choose instance to be queried
12: $\quad y_i^j = \textsc{Query}(\mathcal{O}, x_i^j)$ ▷ query the label to the oracle
13: $\quad \mathcal{F}^k = \textsc{TrainDetector}(D^k \cup y_i^j)$
14: $\quad \overrightarrow{r}^j = \textsc{EstimatePayoff}(\mathcal{F}_{t-1}^k, \mathcal{F}_t^k)$
15: $\quad t = t + 1$
16: **end while**

### 3.5 ALBA Algorithm

Algorithm 1 details the full ALBA algorithm. On lines 4-7 the action set is instantiated by first clustering each dataset in $\mathcal{M}$ into $C$ clusters, and an initial anomaly detector is trained for each dataset. On line 9 the payoff vector that stores the obtained rewards, is initialized to zero. Lines 10-17 contain ALBA's active learning loop. ALBA proceeds in five steps: (i) it selects the group of instances from which to query a label according to the current MAB reward estimate (line 11), (ii) it selects an instance from that group to query (line 12), (iii) it queries the instance's label to the oracle (line 13), (iv) it retrains the model (line 14), and (v) it computes the actual labeling payoff using Eq. 1 or 2 to update the MAB reward estimate for the selected group in step (i) (line 15).

ALBA is computationally more time-efficient in the multi-domain setting than some AL strategies, such as uncertainty sampling. The cost of retraining the detector after an instance has been labeled is identical for both ALBA and every AL technique. However, while most AL techniques, such as uncertainty sampling, use heuristics to estimate the potential labeling payoff of each instance upfront (resulting in $\sum_{k=1}^{K} |D^k|$ computations each query round), ALBA only has to keep track of the labeling payoff of the different groups. This results in only $K \times C$ computations in total per query round.

## 4 Related Work

Table 1: Classification of the *problem dimensions* tackled in AL related work. A check mark (✓) and dash (-) signify what was part of the original problem description.

| | **No. tasks** | | **No. datasets** | | **No. classes** | | **No. views** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Reference** | 1 | $\geq 2$ | 1 | $\geq 2$ | 2 | $\geq 3$ | 1 | $\geq 2$ | **Paradigm** |
| [20,24] | ✓ | - | ✓ | - | ✓ | - | ✓ | - | Classic AL |
| [17,1,28] | - | ✓ | ✓ | - | ✓ | - | ✓ | - | Multi-task AL |
| [29,22,19,9] | ✓ | - | ✓ | - | - | ✓ | ✓ | - | Multi-class AL |
| [26] | ✓ | - | ✓ | - | ✓ | - | - | ✓ | Multi-view AL |
| [14,31] | ✓ | - | - | ✓ | ✓ | - | ✓ | - | MDAL |
| ALBA | ✓ | - | - | ✓ | ✓ | - | ✓ | - | MDAL |

### 4.1 Active Learning

This work only considers sequential, pool-based active learning, where labels are queried one-by-one until the budget is spent. To see how our work fits within the vast body of research on AL, we roughly divide the spectrum of AL techniques along four axis: the *number of tasks* solved, the *number of datasets* considered, the *number of classes to predict* in each dataset, and whether multiple classifiers are learned on *different views* (i.e., feature subsets) of the data. Table 1 summarizes how the related works discussed below, fit these four axis. The classic AL techniques, such as uncertainty sampling [13], query-by-committee [21], expected-error reduction [18], density-based approaches [20], were originally designed for single-task, single-dataset scenario's with the features treated as a single set (view). In-depth surveys on these AL techniques are [20,24].

ALBA differs from these classic AL techniques in two important ways. First, ALBA is explicitly designed for multiple datasets, each of which requires the training of a separate classifier (or anomaly detector). Second, ALBA's use of an MAB strategy fundamentally changes how an acquired label informs subsequent query rounds, because it tracks the marginal gain of acquiring a label of groups of instances and not of single instances. An instance's "true labeling payoff" can *only* be measured by comparing classifier performances before and after retraining with said labeled instance. ALBA can measure this payoff exactly while the classic AL techniques have to resort to heuristics to estimate it upfront. To see why, consider the order of operations. In classic AL we (1) estimate the payoff of getting each instance's label, (2) get the label, and (3) retrain the model. In contrast, ALBA (1) selects a group of instances from a dataset according to the current reward estimate, (2) selects an unlabeled instance from the chosen group, (3) gets the instance's label, (4) retrains the model, and then (5) computes the actual payoff to update the reward estimate for the selected group in step (1).

Most related to our work, are [14] and [31]. [14] proposes a method for MDAL for classification. Their work properly conforms with the MDAL setting. How-

ever, their AL method is fully integrated with the underlying SVM classifier and geared towards text classification. This makes it difficult to use for our experiments without significant alterations to the proposed method. [31] designed a strategy for active learning for multi-domain recommendation. Recommendation is quite different from classification as the optimization target is distinct, rendering the proposed approach unsuitable for our task. The online Appendix 7.1 to this paper provides further details on how our work relates and differs from multi-task, multi-view, and multi-class active learning.

### 4.2 MAB Strategies and Active Learning

Since there is no single best AL algorithm [5], some researchers look at *learning active learning* [10,11]. The idea is to learn how to select the best AL strategy from a pool of strategies, potentially using an MAB approach [10,5], or learning which instances in a dataset are likely to improve the classifier [11]. ALBA differs from all these approaches as it is not concerned with finding the best AL strategy among $K$ strategies for 1 dataset, but rather with identifying the labeling payoff of $K$ different datasets using 1 strategy.

The work of [8] proposes the use of MAB strategies to sequentially select instances presented to the oracle. The main difference with ALBA is that they conceptually view each learned hypothesis as an action, while ALBA equates each action with a group of instances. Moreover, ALBA works for multiple datasets that require potentially different classifiers (hypotheses). Finally, Fang et al. also use an MAB approach to decide which instance to query [6]. There are two key differences with our work. First, ALBA equates *actions* with groups of instances and not with different learned tasks. Second, ALBA explicitly accounts for the diminishing payoffs of labeling additional instances.

## 5 Experiments

We evaluate multi-domain active learning in the context of *anomaly detection* where we have access to real-world multi-domain data consisting of nearly four years of water consumption data from 7 different retail stores. Anomaly detection naturally fits this paper's problem setting for two reasons. First, many real-world anomaly detection problems consist of multiple distinct datasets where an anomaly detector has to be learned for each dataset. Second, while anomaly detection problems typically were posed as unsupervised problems due to difficulties of obtaining labeled data, there is growing evidence that in practice achieving good performance requires labeling some data data [25].

We try to answer following questions empirically:

**Q1.** Does ALBA outperform the classic AL baselines when dealing with multi-domain datasets where $K > 1$?
**Q2.** How does the choice of the action set $\mathcal{A}$ impact ALBA's performance?
**Q3.** How do the choice of the MAB reward function and the query selection strategy impact ALBA's performance?

### 5.1 Experimental Setup[4]

***Compared Methods*** For all approaches, we use the same semi-supervised anomaly detector: SSDO with the Isolation Forest algorithm as prior to generate the unsupervised anomaly scores [25]. One could use other detectors in theory. The unsupervised prior of SSDO allows us to exploit information in the unlabeled data. For active learning, we compare to uncertainty and random sampling as they (1) can be used with any underlying detector, and (2) have been shown to consistently perform well against more complicated AL strategies [9,24].

We compare ALBA to seven baselines, divided into two categories. Category 1 baselines **combine all the datasets into one big dataset and learn a single anomaly detector** using the random (C-RAND) or uncertainty sampling (C-UC) active learning strategy. Though necessary baselines [9], learning a single detector for the combined datasets is likely suboptimal as each dataset has distinct marginal and conditional distributions. This would result in difficult-to-learn regions in the combined instance space for the detector.

Category 2 baselines **treat each domain independently and learn a separate model for each one.** I-U learns a completely unsupervised anomaly detector for each dataset. I-RAND or I-UC use the random and uncertainty sampling strategies in the following way in each query round. First, they apply their AL strategy to each dataset to select the most informative instance within each dataset. Given this set of identified instances, they again apply their respective AL strategy to select the single most informative instance to be labeled. I-RAND and I-UC do not attempt to ensure any balance in terms of how many instances are queried from a given a dataset. Finally, I-R-RAND or I-R-UC impose an additional *restriction* on I-RAND or I-UC. Given a fixed total query budget $T$ and $K$ datasets, at most $\lfloor T/K \rfloor$ instances can be sampled from a given dataset.

***Evaluation Metrics*** The performance of the *anomaly detector* on a single dataset is evaluated using the *area under the receiver operating characteristic* (AUROC) as is standard in anomaly detection [3]. The performance on a multi-domain dataset is obtained by averaging the AUROC scores on the $K$ individual datasets ($\text{AUROC}_K$). The performance of an *active learning* strategy is revealed by the progress curve which captures how the $\text{AUROC}_K$ evolves as a function of the number of labeled instances (i.e., the spent label budget) [24]. As the number of experiments increases, these curves are summarized using the *area under the active learning curve* (AULC) [24]. AULC scores are $\in [0, 1]$ and higher scores are better.

***Benchmark Data*** We use 7 real-world datasets that each track the water consumption of a different retail store, measured every 5 minutes, over the course of 4 years. Each of the 7 datasets is further divided into 24 datasets by grouping the

[4] Online Appendix 7.2 has detailed information on the (choice of) evaluation metrics, benchmark data, and hyperparameters. It also has additional results on the impact of the dataset characteristics on ALBA's performance.

data per hour-of-the-day, yielding 7 (stores) $\times$ 24 (hours) = 168 fully labeled water datasets. This division is necessary as the hour-of-the-day strongly influences the observed water consumption. The binary labels are "normal" or "anomalous" usage (e.g., water leaks). Then, we transform each hour-long segment into a feature-vector[5], and train a separate anomaly detector per dataset.

We now construct an appropriate multi-domain AL benchmark as follows. First, we compute for each of the 168 datasets a *labeling payoff score* which is simply the difference in AUROC obtained by (1) SSDO trained with 20% of the data labeled and (2) SSDO trained without labels. Then, we construct a *multi-domain dataset* by selecting $K$ datasets from the set of 168 water datasets. A fraction $\psi$ of these $K$ datasets are selected to have a high labeling payoff score, while the remaining datasets $(1 - \psi K)$ have a low labeling payoff scores. By varying $K$ in $[2, 10]$ and $\psi$ in $[0.1, 1]$, we obtain the full benchmark of 54 unique multi-domain datasets.

***Setup*** Given a multi-domain dataset $\mathcal{M}$ from the benchmark, the anomaly detector, and an AL method, each experiment proceeds in four steps. First, each of the $K$ datasets in $\mathcal{M}$ is randomly divided into 2/3 train and 1/3 test set. Second, we simulate an oracle iteratively labeling one training instance at a time selected by the AL method across the $K$ datasets, until the label budget $T = 500$ is spent. Third, each iteration, the appropriate anomaly detector is retrained and we recompute the $\text{AUROC}_K$ on the test data. Each experiment is repeated 5 times to average out any random effects, resulting in a total of 7 (methods) $\times$ 5 $\times$ 54 = 1890 experiments where each experiment has $T + K$ training and $T \times K$ evaluation runs. The baselines have no hyperparameters, ALBA has three. These are $C = 5$ using KMEANS, the SWA bandit algorithm with the *cosine* reward function, and a RAND query selection strategy.

### 5.2 Experimental Results

***Q1:* ALBA *versus the baselines*** Figure 1 plots the progress curves for 5 multi-domain datasets randomly selected from the benchmark of 54 datasets. The plots[6] reveals four insights. One, all approaches (except C-UC) outperform the unsupervised baseline after about 100 query rounds. Two, learning a separate anomaly detector per dataset clearly outperforms combining all datasets and learning a single detector, as evidenced by the lagging performances of C-RAND and C-UC versus the other baselines. Three, a completely random AL strategy surprisingly outperforms the heuristic strategy in the multi-domain setting, as evidenced by C-RAND, I-RAND, and I-R-RAND outperforming respectively C-UC, C-RAND, and C-R-RAND on all but one benchmark dataset. Four, I-RAND and I-R-RAND perform similarly.

[5] We use 8 statistical (average, standard deviation, max, min, median, sum, entropy, skewness, and Kurtosis) and 2 binary features (whether its a Friday or a Sunday), 10 in total.

[6] See online Appendix 7.3 for the plots for all 54 benchmark datasets.

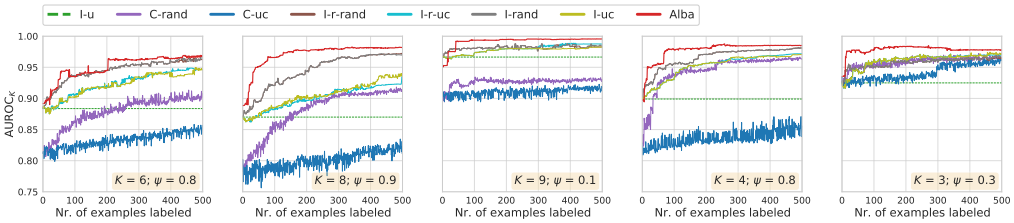


Fig. 1: The figure shows the progress curves of ALBA and the baselines for 5 multi-domain datasets randomly selected from the full benchmark. Each progress curve shows how the AUROC$_K$ evolves as a function of the total number of instances labeled by the oracle. The characteristics of each selected multi-domain dataset ($K$ and $\psi$) are shown on the corresponding plot.

After enough query rounds, all approaches (except the unsupervised baseline) will converge to the performance of a fully supervised classifier. Better AL strategies, however, converge faster. To investigate each method's convergence, we look at their performance after both 100 query rounds and 500 query rounds. Table 2 shows for the 54 benchmark multi-domain datasets how many times ALBA wins/draws/loses in terms of AULC versus each baseline[7] as well as the average AULC rank for each method [4]. Table 2a shows the results after 100 queries, while Table 2b shows the results after 500 queries. After 100 query rounds, the Friedman test rejects the null-hypothesis that all methods perform similarly (p-value $<$ 1e-8). The post-hoc Bonferroni-Dunn test [4] with $\alpha = 0.05$ finds that ALBA is significantly better than every baseline. After 500 query rounds, ALBA is still significantly better than most baselines, except I-RAND and I-R-RAND. This illustrates that the methods start converging. However, ALBA converges faster (this is also illustrated by the progress curves), which is especially useful in scenario's where labeling is costly, such as anomaly detection.

***Q2: Impact of the Choice of Action Set*** We explore the effect of the granularity of the action set considered by ALBA on its performance. We do so by varying the number of clusters $C$ per dataset. When $C = 1$, the action set is coarse-grained as each action corresponds to a full dataset. As $C$ increases, the action set becomes more fine-grained as each dataset is further partitioned into groups using KMEANS. Figure 2 points to a correlation between $C$ and ALBA's performance. As $C$ increases, the MAB method can make a more fine-grained estimate of the usefulness of different groups of instances.[8]

***Q3: Impact of the MAB Algorithm, Reward Function, and Query Selection Strategy*** We explore the effect of the choice of MAB algorithm, reward

[7] All the experimental evaluations maintain a precision of 1e-4 and a threshold of 0.001 (e.g., to determine the similarity of two AULC scores).

[8] See online Appendix 7.3 for a more detailed discussion.

Table 2: The table shows the number of AULC wins/draws/losses of ALBA versus each baseline, and the average AULC rank (± Standard Deviation) of each method on the full benchmark, both after 100 and 500 query rounds.

| | Nr. of times ALBA: | | | Ranking |
|---|---|---|---|---|
| **Method** | **wins** | **draws** | **loses** | **Avg. ± SD** |
| ALBA | - | - | - | **1.315 ± 0.894** |
| I-RAND | **48** | 2 | 4 | 2.639 ± 0.573 |
| I-R-RAND | **48** | 2 | 4 | 2.639 ± 0.573 |
| I-U | **48** | 2 | 4 | 4.333 ± 1.656 |
| I-UC | **53** | 0 | 1 | 5.157 ± 0.551 |
| I-R-UC | **53** | 0 | 1 | 5.231 ± 0.497 |
| C-RAND | **54** | 0 | 0 | 6.741 ± 0.865 |
| C-UC | **54** | 0 | 0 | 7.944 ± 0.404 |

(a) Results @ 100 query rounds

| | Nr. of times ALBA: | | | Ranking |
|---|---|---|---|---|
| **Method** | **wins** | **draws** | **loses** | **Avg. ± SD** |
| ALBA | - | - | - | **1.306 ± 0.710** |
| I-RAND | **45** | 2 | 7 | 2.417 ± 0.507 |
| I-R-RAND | **46** | 1 | 7 | 2.417 ± 0.507 |
| I-R-UC | **54** | 0 | 0 | 4.537 ± 0.686 |
| I-UC | **53** | 1 | 0 | 4.546 ± 0.512 |
| I-U | **54** | 0 | 0 | 6.370 ± 0.818 |
| C-RAND | **54** | 0 | 0 | 6.491 ± 0.717 |
| C-UC | **54** | 0 | 0 | 7.917 ± 0.382 |

(b) Results @ 500 query rounds

function, and query selection strategy on ALBA's performance. Table 3 shows the resulting average AULC ranks ($C = 5$). The best-performing version of ALBA uses a *cosine* reward function and *random* instance selection strategy. Generally, the RAND versions of ALBA outperform their UC counterparts $\sim 63\%$ of time on the full benchmark, and they draw $\sim 18\%$ of the time. Repeating the analysis with $C = 1$, the RAND versions outperform their UC counterparts $\sim 88\%$ of time and they draw $\sim 5.5\%$ of the time. This aligns with the theoretical results. When fixing the query strategy, our proposed cosine reward function outperforms the entropy reward function. See online Appendix 7.3 for linear regression analyses on these results, as well as further analyses on the impact of the dataset characteristics, $K$ and $\psi$, on the performance of ALBA.

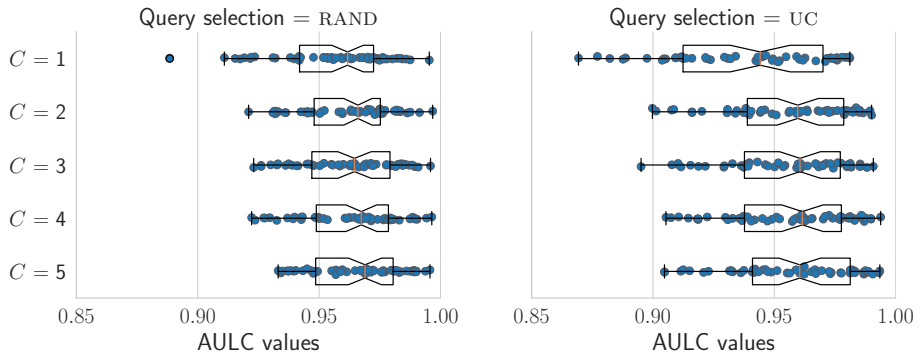


Fig. 2: Box plot overlaying a scatter plot of the AULCs obtained by ALBA with different values for $C$ on the 54 benchmark datasets. Results are shown for two versions of ALBA (with different instance selection strategies).

Table 3: Average AULC rank (± Standard Deviation) of versions of ALBA with different settings for the query selection strategy and reward function ($C = 5$).

| **Reward function** | **Query sel. strategy** | **Ranking Avg. ± SD** |
|---|---|---|
| cosine | rand | **1.806 ± 0.813** |
| cosine | uc | 2.944 ± 0.926 |
| entropy | rand | 2.241 ± 0.843 |
| entropy | uc | 3.009 ± 0.825 |

## 6 Conclusion

This paper tackled the multi-domain active learning problem for anomaly detection, which often arises in practice. The key challenge was to determine from which dataset an instance should be queried as labels are not equally beneficial in all domains. To cope with this problem, we proposed a method (ALBA) that exploits multi-armed bandit strategies to track the label-informativeness of groups of instances over time and decides which instances are optimal to query to an oracle. Empirically, ALBA outperformed existing active learning strategies on a benchmark of 7 real-world water consumption datasets.

**Acknowledgements** This work is supported by the Flemish government "Onderzoeksprogramma Artificiële Intelligentie Vlaanderen", "Agentschap Innoveren & Ondernemen (VLAIO)" as part of the innovation mandate HBC.2020.2297, the FWO-Vlaanderen aspirant grant 1166222N, and Leuven.AI, B-3000 Leuven, Belgium.

## References


1. Acharya, A., Mooney, R.J., Ghosh, J.: Active multi-task learning using both latent and supervised shared topics. In: Proceedings of the 2014 SIAM International Conference on Data Mining. pp. 190–198 (2014)
2. Bubeck, S., Cesa-Bianchi, N., et al.: Regret analysis of stochastic and nonstochastic multi-armed bandit problems. Foundations and Trends® in Machine Learning **5**(1), 1–122 (2012)
3. Campos, G.O., Zimek, A., Sander, J., Campello, R.J., Micenková, B., Schubert, E., Assent, I., Houle, M.E.: On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Mining and Knowledge Discovery **30**(4), 891–927 (2016)
4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine learning research **7**, 1–30 (2006)
5. Desreumaux, L., Lemaire, V.: Learning active learning at the crossroads? evaluation and discussion. arXiv preprint arXiv:2012.09631 (2020)
6. Fang, M., Tao, D.: Active multi-task learning via bandits. In: Proceedings of the 2015 SIAM International Conference on Data Mining. pp. 505–513 (2015)
7. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. Journal of Machine Learning Research **17**(1), 2096–2030 (2016)
8. Ganti, R., Gray, A.: Building bridges: Viewing active learning from the multi-armed bandit lens. arXiv preprint arXiv:1309.6830 (2013)
9. He, R., He, S., Tang, K.: Multi-domain active learning: A comparative study. arXiv preprint arXiv:2106.13516 (2021)
10. Hsu, W.N., Lin, H.T.: Active learning by learning. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence (2015)
11. Konyushkova, K., Sznitman, R., Fua, P.: Learning active learning from data. arXiv preprint arXiv:1703.03365 (2017)
12. Levine, N., Crammer, K., Mannor, S.: Rotting bandits. Advances in Neural Information Processing Systems **30** (2017)
13. Lewis, D.D., Catlett, J.: Heterogeneous uncertainty sampling for supervised learning. In: Machine Learning Proceedings, pp. 148–156. Morgan Kaufmann (1994)
14. Li, L., Jin, X., Pan, S.J., Sun, J.T.: Multi-domain active learning for text classification. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1086–1094 (2012)
15. Perini, L., Vercruyssen, V., Davis, J.: Class prior estimation in active positive and unlabeled learning. In: Proceedings of the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI 2020). pp. 2915–2921. IJCAI-PRICAI (2020)
16. Perini, L., Vercruyssen, V., Davis, J.: Transferring the contamination factor between anomaly detection domains by shape similarity. In: Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (2021)
17. Reichart, R., Tomanek, K., Hahn, U., Rappoport, A.: Multi-task active learning for linguistic annotations. In: Proceedings of ACL-08: HLT. pp. 861–869 (2008)
18. Roy, N., McCallum, A.: Toward optimal active learning through monte-carlo estimation of error reduction. Proceedings of the 18th International Conference on Machine Learning pp. 441–448 (2001)
19. Sener, O., Savarese, S.: Active learning for convolutional neural networks: A core-set approach. arXiv preprint arXiv:1708.00489 (2017)

20. Settles, B.: Active learning. Synthesis Lectures on Artificial Intelligence and Machine Learning **6**(1), 1–114 (2012)
21. Seung, H., Opper, M., Sompolinsky, H.: Query by committee. In: Proceedings of the 5th Annual Workshop on Computational Learning Theory. pp. 287–294 (1992)
22. Sinha, S., Ebrahimi, S., Darrell, T.: Variational adversarial active learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5972–5981 (2019)
23. Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika **25**(3/4), 285–294 (1933)
24. Trittenbach, H., Englhardt, A., Böhm, K.: An overview and a benchmark of active learning for outlier detection with one-class classifiers. Expert Systems with Applications **168**, 114372 (2021)
25. Vercruyssen, V., Meert, W., Verbruggen, G., Maes, K., Bäumer, R., Davis, J.: Semi-supervised anomaly detection with an application to water analytics. In: Proceedings of the IEEE International Conference on Data Mining. pp. 527–536 (2018)
26. Wang, W., Zhou, Z.H.: On multi-view active learning and the combination with semi-supervised learning. In: Proceedings of the 25th International Conference on Machine learning. pp. 1152–1159 (2008)
27. Wei, K., Yang, Y., Zuo, H., Zhong, D.: A review on ice detection technology and ice elimination technology for wind turbine. Wind Energy **23**(3), 433–457 (2020)
28. Xiao, Y., Chang, Z., Liu, B.: An efficient active learning method for multi-task learning. Knowledge-Based Systems **190** (2020)
29. Yoo, D., Kweon, I.S.: Learning loss for active learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 93–102 (2019)
30. Zhang, Y., Yang, Q.: A survey on multi-task learning. arXiv preprint arXiv:1707.08114 (2017)
31. Zhang, Z., Jin, X., Li, L., Ding, G., Yang, Q.: Multi-domain active learning for recommendation. In: 30th AAAI Conference on Artificial Intelligence (2016)