Instance-Level Metalearning for Outlier Detection

Long Vu , **Peter Kirchner** , **Charu C. Aggarwal** , **Horst Samulowitz**
IBM Research
lhvu@us.ibm.com, pdk22@cornell.edu, {charu, samulowitz}@us.ibm.com

Abstract

A machine learning task can be viewed as a sequential pipeline of different algorithmic choices, including data preprocessing, model selection, and hyper-parameter tuning. Automated machine learning selects this sequence in an automated manner. While such approaches are natural in supervised settings, they remain challenging for unsupervised tasks such as outlier detection because of the lack of availability of label-centric feedback. In this paper, we present an instance-level metalearning approach for outlier detection. This approach learns how outlier instances are related to normal points in many labeled data sets to create a supervised meta-model. This meta-model is then used on a new (unlabeled) data set to predict outliers. We show the robustness of our approach on several benchmarks from the OpenML repository.

1 Introduction

Machine learning tasks contain multiple choices across various stages of data processing, such as imputation, scaling, model selection, and hyper-parameter setting. Each such sequence of choices is referred to as a pipeline. The goal of automated machine learning is to optimize the pipeline choice for the specific data set and task at hand. Although automated machine learning is popular in the supervised setting [Bouneffouf *et al.*, 2020; Feurer *et al.*, 2015; Franceschi *et al.*, 2017; He *et al.*, 2021; Hutter *et al.*, 2011; Lai *et al.*, 2021; Lee *et al.*, 2019; Li *et al.*, 2018; Li *et al.*, 2008; Vartak *et al.*, 2017; Wistuba *et al.*, 2015; Yao *et al.*, 2018], only a few methods exist for unsupervised tasks such as clustering and outlier detection [Bahri *et al.*, 2022; Burnaev *et al.*, 2015; Fan *et al.*, 2019; Mourer *et al.*, 2023; Singh and Vanschoren, 2022; Zhao *et al.*, 2021]. Unsupervised settings are also challenging because of the lack of label-centric feedback.

A key insight is that even though unsupervised settings do not enable learning from *new* data sets, one can still learn about the types of pipeline choices that work well in other data sets where supervision is available. Such supervised settings correspond to imbalanced classification problems in which rare classes are treated as outliers. Rare class instances often exhibit typical structural relationships to the full data set, which can be captured as a supervised meta-model and leveraged for outlier identification in a new data set. This *instance-level* knowledge that captures structural relationships between instances and full data (for both outliers and non-outliers) can be transferred into a model that is used to identify outliers in a new, unlabeled data set. This approach is referred to as *metalearning* — however in the literature metalearning often transfers *dataset-level* knowledge [Zhao *et al.*, 2021], whereas the approach in this paper transfers *instance-level* knowledge. The key advantage of instance-level knowledge is that it is able to learn important *local* characteristics of the data that are salient for outlier detection. Such characteristics create a much more refined model for metalearning. The work is also closely related to outlier ensembles [Aggarwal, 2013; Aggarwal and Sathe, 2015; Aggarwal and Sathe, 2017] because it does not identify a single pipeline for outlier identification but uses a combination function of the scores from different pipelines in an instance-specific way. This combination function of scores from different pipelines is learned in the form of a meta-model that is constructed using the knowledge gained from external data sets. Therefore, the contributions of this paper are as follows:

- We propose an instance-level metalearning approach, referred to as **T-AutoOD** for outlier detection. This meta-model is learned using a labeled resource of classification data sets and it builds a model of how the outputs from different pipelines should be combined in order to obtain the most accurate outlier score.
- It is shown how this learned model can be used to find outliers on a new unlabeled data set, which essentially creates a supervised meta-model for ensembling different pipelines in an instance-specific way.
- Experimental results are presented showing the competitiveness of the approach over existing baselines.

This approach is a form of transfer learning [Pan and Yang, 2009] from imbalanced classification data sets (i.e., supervised anomaly detection data sets), which are copiously available. This collection is referred to as the *transfer resource*. Since the transfer resource is used to learn how the scores from different pipelines are combined to create a unified outlier score, it can be viewed as a *metalearning ensemble*. In this sense, this work combines ideas from metalearning, transfer learning, and outlier ensembles.

1.1 Related Work

Outlier analysis [Aggarwal, 2017; Chandola *et al.*, 2009] has been widely studied in the machine learning literature, and it is used as an alternative to imbalanced classification when labels are not available. An important challenge in outlier detection (and other unsupervised tasks) is that of model selection since labels are not available for data-driven feedback. In contrast, a vast array of model selection and tuning methods are available for the supervised setting, including Bayesian hyperparameter optimization [Feurer *et al.*, 2015], gradient-based optimization [Franceschi *et al.*, 2017], model-based optimization [Hutter *et al.*, 2011], bandit methods [Li *et al.*, 2018], and sparse models [Liuliakov *et al.*, 2023]. Automated machine learning methods have also been proposed in cold-start [Lee *et al.*, 2019; Vartak *et al.*, 2017] and zero-knowledge settings [Real *et al.*, 2020]. Surveys on automated machine learning may be found in [Bouneffouf *et al.*, 2020; He *et al.*, 2021; Yao *et al.*, 2018].

In spite of the outstanding success of automated machine learning methods in the supervised setting, the performance of such methods in unsupervised problems has been quite modest. for example, model selection in clustering is typically addressed using heuristic measures [Fan *et al.*, 2019; Heller and Ghahramani, 2005; Mourer *et al.*, 2023; Vaithyanathan and Dom, 1999]. The challenges associated with model selection in anomaly detection have been discussed in [Aggarwal and Sathe, 2017], and only a few methods exist for model selection in anomaly detection [Bahri *et al.*, 2022; Burnaev *et al.*, 2015; Zhao *et al.*, 2021]. The work that is closest to ours (in terms of goals) is **MetaOD** [Zhao *et al.*, 2021], which is also a meta-learning framework for outlier detection. However, **MetaOD** works by evaluating the performance of a candidate model on similar historical tasks based on *dataset-level* meta-features. This is different from our approach, which captures structural characteristics with *instance-level* features. The use of outlier scores to represent data sets has been pursued in the context of supervised learning [Micenková *et al.*, 2014], although its use for unsupervised meta-learning at the instance level is non-obvious. This type of instance-specific metalearning is also closely related to transfer learning [Pan and Yang, 2009], and it is better targeted to exploiting the local characteristics of data sets compared to dataset-level metalearning.

2 Algorithmic Details of T-AutoOD

This section presents an overview of the **T-AutoOD** framework along with specific details. The first step of **T-AutoOD** is to create the transfer resource from existing classification data sets. The data sets of the transfer resource are then converted into a single large data set of a particular dimensionality. The goal of converting the data sets into a uniform representation is to (i) ensure learnability across data sets, and (ii) capture the same point-specific structural characteristics across data sets that are indicative of anomalous behavior. This homogenized representation is then carried over to the (new) unlabeled data set, where a classification algorithm (trained on the transfer resource) is used to perform the anomaly prediction on a new data set (without labels).

Algorithm 1 T-AutoOD-Train

```
Input: Transfer Resource of datasets $\mathcal{D}_T$
Output: Transfer Model: $\mathcal{M}$
1: Generate pipelines $\mathcal{P}$.
2: Apply each $p_i \in \mathcal{P}$ to each data set $D_j \in \mathcal{D}_T$.
3: Create a single data set $\mathcal{D}$ with rows corresponding to
   instances in all $D_j \in \mathcal{D}_T$. Each row contains anomaly
   scores from all $p_i \in \mathcal{P}$ on that instance and its label.
4: Create supervised classification model $\mathcal{M}$ on data set $\mathcal{D}$.
5: return $\mathcal{P}$, $\mathcal{M}$
```

Algorithm 2 T-AutoOD-Test

```
Input: Model $\mathcal{M}$, Pipelines: $\mathcal{P}$, Unlabeled Data Set $U$
Output: Anomaly scores of $U$
1: Apply pipelines in $\mathcal{P}$ to each instance in $U$ to create
   score-based representation $U'$ of $U$.
2: Apply supervised model $\mathcal{M}$ to each instance in $U'$.
3: return classification scores generated by $\mathcal{M}$
```

The overall process of transfer learning in the proposed **T-AutoOD** algorithm is shown in Algorithms 1 and 2. A set of (labeled) anomaly detection data sets is created as a resource for transfer learning. These data sets are generated from existing classification data sets by subsampling the minority class if necessary to achieve the desired contamination. The data sets in the transfer resource are used to generate the anomaly scores for candidate outlier detection pipelines in an unsupervised manner. Each pipeline yields an anomaly score, and these different scores are used to create features. The features indirectly capture different structural relationships of the specific data point (indicating anomalous behavior) to the data set at hand. Such an approach creates a homogenized feature representation across all data sets and therefore a single large data set can be created. The basic principle underlying this feature representation is that particular types of anomaly score patterns recur across different data sets, and these patterns can be learned for the transfer resource because anomaly labels are available.

This idea motivates the design of a supervised model, in which the relationship between the generated features and the true anomaly labels (from the knowledge transfer resource) can be learned using a classification model. This paper uses a light gradient boosted machine (LGBM) classifier because of its combination of accuracy and efficiency. For a new (unlabeled) data set, the features are generated using the same approach and passed through this supervised model for prediction.

We assume that a total of p labeled data sets are available as the transfer resource. Each point is tagged with a binary label depending on whether it is an anomaly. The ith data set is assumed to contain n_i data points. In some cases, these data sets are prepared from existing classification data sets by sparsifying one or more classes. The details of the preparation of the transfer resource will be provided in a later section. After preparing the transfer resource, the following steps are used:

1. **Unsupervised pipeline training on transfer resource:** A number of unsupervised pipelines are devised from known outlier detection algorithms, such as the k-nearest neighbor method, LOF, isolation forest, and randomized hashing [Sathe and Aggarwal, 2016]. Each pipeline consists of a specific set of feature imputation/scaling steps, feature engineering step, algorithm selection and hyper-parameter setting. It is assumed that a total of m pipelines are available, and therefore m anomaly scores are output for each data point (one for each pipeline). Although this step is time consuming (since each pipeline needs to be run on each data set), it needs to be done only once up front during the transfer learning process.

2. **Creating an anomaly score-based representation of the transfer resource:** The aforementioned step creates a feature representation of m anomaly scores from each of the $N = \sum_{i=1}^{p} n_i$ data points across the p different data sets in the transfer resource. This homogenized representation across different data sets in the transfer resource can be used to create a single $N \times m$ data set of anomaly scores. This representation can be enriched with additional features and we discuss one such feature later in this paper.

3. **Learning a transfer model from the score-based representation:** The aforementioned $N \times M$ data set is then trained with a classifier by using the anomaly labels available with the points in the transfer resource. In principle, any off-the-shelf classifier could be used, although the light gradient boosting machine (LGBM) classifier was found to be a good choice because of its combination of accuracy and efficiency. The resulting binary classification model captures *how different patterns of anomaly scores* relate to the probability of a point being a true anomaly across different data sets in the transfer resource. This learned model indirectly captures the relationship between the different anomaly-specific structural patterns of data points in the transfer resource and anomaly labels.

4. **Applying learned transfer model to anomaly detection on a new data set:** The learned model can easily be used for prediction on a new data set. For each point in the new data set, the features are constructed in the same manner as is the case for the data points in the transfer resource. The learned LGBM model is then used to predict the binary class label (i.e., the label of whether or not the point is anomalous) of the new data point. Such an approach effectively learns the patterns of anomaly scores that are most indicative of anomalous behavior across different data sets.

The above description provides an overview of the **T-AutoOD** framework. In the following, the details of individual steps will be provided. Furthermore, since the transfer resource is an important part of the learning process, its creation will be discussed first.

2.1 Creation of Transfer Resource from Labeled Data Sets

Existing data sets for classification can be used for transfer learning in anomaly detection as long as the underlying data sets have imbalanced class distributions. This is because the rare class can be treated as anomalous, whereas the remaining data set can be treated as the normal class. In fact, the main difference between rare class detection and outlier detection is that observed labels are not present in the latter case. Even when the underlying data sets are not balanced, one of the classes can be downsampled to artificially create an anomalous class. Such an approach of creates multiple derived data sets from a single resource, which greatly augments the training data for transfer learning.

The base data sets used for creating the transfer resource were drawn from classification data sets in OpenML[1]. In order to create each derived data set, we assigned the outlier label to the minority class and the inlier label to the remaining classes. We adjusted derived data set class imbalance (one-versus-rest) as necessary to achieve target outlier frequency of 1% by down-sampling. Derived data sets with fewer than 10 outliers were excluded. A maximum of 50 columns was used, and multiple derived data sets (with non-overlapping columns) were created for base data sets with greater than 50 columns. To avoid label leakage, data sets originating from a given base data set appeared exclusively either in the labeled transfer resource or in the unlabeled testing set but never in both.

This approach was used to first create 1200 data sets. We then used Isolation Forest and Average KNN, outlier detection algorithms in *scikit-learn* open-source package[2] to train and score these 1200 data sets. For each data set, we then calculated two AUC ROC scores by Isolation Forest and Average KNN, using the ground-truth labels in the original OpenML classification data sets. If both AUC ROC scores were greater than 0.5, indicating that the data set was indeed an outlier detection data set, the data set was kept. Otherwise, it was discarded. Finally, there were 520 data sets that passed the AUC ROC score check and were used in the transfer resource for our method. Note that only a subset of these 520 data sets were used in the transfer resource, because some of the data sets were reserved for testing (as discussed in the evaluation section).

2.2 Running Pipelines on Transfer Resource

After the transfer resource was created, unsupervised pipelines were run on the data sets. A pipeline includes steps to preprocess the data set (including imputing for missing values and scaling) as well as steps for feature engineering and final outlier scoring. Data imputation was necessary because many of the base data sets used for creating the transfer resource were missing entries. The sequence of steps in a particular pipeline can be summarized as follows:

$$\text{Impute} \Rightarrow \text{Scale} \Rightarrow \text{Feature-Engineer} \Rightarrow \text{Score}$$

[1]https://www.openml.org

[2]https://scikit-learn.org

Note that each choice affects outlier scores of individual points in the data set, and the goal of metalearning is to identify the relationship between pipelines and anomaly scores. What are the choices for each of the aforementioned steps? The implementation derived these choices from the base API `sklearn.pipeline.Pipeline` of (*scikit-learn*). For example, the choices for imputation and scaling are available as off-the-shelf APIs in *scikit-learn*. The imputation step uses either a simple imputer or an iterative imputer. This pipeline step is not needed for fully specified data sets in which no values are missing.

The scaling step also allows several choices. It could (i) standardize the data to zero mean and unit variance, (ii) divide an attribute by its maximum absolute value in the data set, (iii) map an attribute to the $[0, 1]$ interval with min-max scaling, or (iv) perform robust standardization that normalizes to zero mean and unit variance after softening the effect of extreme values in modeling. The details of these choices are also described within the pipeline API of *scikit-learn*. The scaling of the different attributes directly affects the outlier scores in most detectors (e.g., k-nearest neighbor detector) because it regulates the relative importance of the different attributes while calculating aggregate measures such as distances. The feature engineering step adds polynomial features as well as PCA features. The goal of the feature engineering step is to enrich the base representation with transformations of the original attributes.

Finally, the estimation step selects one of a set of robust outlier detection algorithms, which are described in [Aggarwal, 2017]. These outlier detection algorithms have been extensively tested by other researchers and a subset was identified based on available performance comparisons [Aggarwal, 2017; Aggarwal and Sathe, 2017]. The subset of outlier detection algorithms used were the Isolation Forest, Average KNN, ElipticEnvelope[3], LOF, randomized hashing [Sathe and Aggarwal, 2016], one-class-SVM, Copula-based outlier detection [Li *et al.*, 2020], PCA, and PCA-based outlier detection. All these algorithms are described in detail in [Aggarwal, 2017]. The hyper-parameters for the pipelines were chosen at random from a recommended range as suggested in the *scikit-learn* implementations of these algorithms. A specific combination of each of the aforementioned steps (including hyper-parameter choice) defines a pipeline. The application of the pipeline to each data record of the transfer resource generates a numerical outlier score. A total of 400 pipelines were selected up front, which resulted in a total of $m = 400$ outlier scores.

2.3 Transfer Resource to Uniform Representation

As discussed above, a total of m outlier scores are created for each of the n_i points in the ith data set. This process transforms each data point of each data set in the transfer resource to the same m-dimensional representation of outlier scores. Since a total of $N = \sum_{i=1}^{p} n_i$ points exist over all the p data sets, a single master data matrix of size $N \times m$ containing the feature representations of the N points (in the rows of the matrix) can be created. An $N \times 1$ vector of ground-truth binary labels (indicating anomalousness) of the N points is also available. This integrated data set from the transfer resource can be rather large, since each point in each data set of the transfer resource has its own distinct m-dimensional entry in the data set. For example, a total of about 6.4×10^6 records were generated across the different data sets in the transfer resource used in this paper.

The $N \times M$ data matrix along with an $N \times 1$ vector of ground-truth values represents a traditional classification data set. The training process over this data set is described in the next section.

2.4 Training the LGBM on Transfer Resource

Once the transfer resource has been converted to a homogenized score-based representation, it is used for constructing a trained model. There are several choices available for selecting the classifier. It is important to use a classifier that is both accurate and efficient because of the large sizes of the data involved. The LGBM classifier was used because of its combination of accuracy and efficiency. We used an out-of-the-box LGBM classifier[4] with its default hyperparameters in this paper.

However, the classification training was preceded by a phase of feature selection. The first step was to select $d \ll M$ most discriminative features using the XGB model [Chen and Guestrin, 2016]. The value of d used was 20. In order to learn the LGBM model, a holdout method is used for splitting the integrated $N \times d$ data set to learn its parameters. Here, an important observation is that one cannot use the type of random-split as is common in hold-out strategies for classification in which data *points* are split between the two partitions. The integrated data set constructed from the transfer resource is generated from multiple base data sets — splitting the points from a single base data set into the training data set and validation data set (or for creating the test data set) is likely to lead to label leakage between training and test/validation data. Therefore, the splitting strategy always makes sure that points from a single base data set always lie in one part of the split (e.g., training or test). Furthermore, when multiple sparsified data sets are derived from the same data set, all such derived data sets are included in the same partition of the holdout set. The LGBM model was then trained using the training/validation data of the holdout partitions. This trained model is then used for prediction of anomalies on a new data set.

2.5 Prediction on New Unlabeled Data Set

Given a new unlabeled data set, the goal is to find its outliers by using both existing outlier detection algorithm pipelines on the current data set as well as by leveraging the learned model from the transfer resource. To apply the trained LGBM model to predict outliers in a new, unlabeled data set, the data set needs to be transformed to have the same d features used by the LGBM model. The first step is to convert this data set to the same d-dimensional representation on which the LGBM was trained (after feature selection by XGB), which requires the same unsupervised outlier detection algorithms.

[3]API from sklearn: sklearn.covariance. EllipticEnvelope

[4]https://lightgbm.readthedocs.io

# DS	# Samples	Mean Size	Mean Dim.
520	6.4M	12097	29

Table 1: Characteristics of transfer resource

The small (feature selected) value of d ensures that the only a small number of pipelines need to be executed during prediction. This process results in an d-dimensional score-based representation of the data set that is identical to that of the training data both in terms of dimensionality and interpretation of attributes.

The trained LGBM model was used for binary classification of each data point in the unlabeled data set to the normal/anomalous class. The LGBM model is capable of producing a real-valued output indicating propensity to belong to the anomalous class. This real-valued output can be treated as the outlier score of the point, and it is useful for evaluating the algorithm using the ROC AUC measure.

3 Experimental Evaluation

This section presents the experimental evaluation[5] of **T-AutoOD** on a variety of data sets. All data sets were derived from OpenML [6], and were generated from classification data sets (after possible sparsification). The datasets for testing were generated in the same manner as that used for generating the transfer resource. The data sets were divided into training and testing portions while accounting for the fact that multiple derived data sets could be constructed from the same base data set by sparsifying different classes. In order to prevent leakage of labels from training to test data, (i) all data sets that were derived from the same base data set were either placed in the training/validation data or placed in the test data, and (ii) data sets were placed in their entirety in either the training or test data (without splitting a base data set across training and test data).

The overall effect of this approach was that a particular base data set could contribute either to the transfer resource or to the testing data set but not both. This restriction was important because multiple data sets derived from the same base data set were related, which could result in overly optimistic performance results from label leakage. An overview of the characteristics of the derived data sets is given in Table 1. There was also a wide variation in data size and dimensionality across derived data sets, which ensured that the transfer learning process was able to capture the importance of data-specific structural characteristics across diverse resources.

The following baselines were used for our comparison:

- **Oracle:** The oracle approach automatically selects the best pipeline out of $m = 400$ pipelines, for each *test data set* as if it were known a priori. It is noteworthy that it is not possible to implement such an approach, since the test data sets are unlabeled (and the optimal pipeline cannot be known a priori with unlabeled data). However, the oracle approach is important because it provides an upper bound on the performance.
- **Single-best:** The single-best pipeline uses a similar principle as the oracle approach in selecting the best pipeline, except that it does so on the *training* data set (i.e., transfer resource) as a validation procedure. Note that it is actually possible to find the single best pipeline (unlike in the case of the oracle), since the labels of the test data are not used for selection. Furthermore, since the single best pipeline is selected on the training data, it is fixed across the different test data sets. However, the performance of the single-best approach is significantly inferior to the oracle method. Nevertheless, it provides a reasonable baseline for comparison.
- **MetaOD** [Zhao *et al.*, 2021]: This method builds a recommendation system that predicts a ML outlier detection model for an unseen data set. We used their open-source code[7] to find their predicted model and then used the model to find outliers in unseen data sets.

The ROC scores were used as the evaluation measure for the algorithm, as is common in outlier detection. The ROC scores were constructed using the real-valued output of the classifier in combination with the ground-truth values. In particular, we used two benchmarks to evaluate and compare our methods against the aforementioned baselines:

- **Internal benchmark:** We first created 520 outlier detection (i.e., derived) data sets. We then split these 520 derived data sets into a training set of 416 derived data sets and testing set of 104 derived data sets. The split ensured that all derived data sets from the same base data set were put into either training or testing set, but not both. This is done to avoid data leakage from training to test data. Then, our model was trained on the training set and tested on the testing set.
- **External benchmark:** In outlier detection research, ODDS[8] and DAMI[9] data sets have been used extensively for performance evaluation. However, we notice that the outlier ratio in these data sets varies widely from less than 1% to more than 43%. In fact, a large portion of these data sets have an outlier ratio of greater than 10%, which we believe is inappropriately high either in terms of how outliers are naturally defined or in terms of how they are detected in an unsupervised manner. We decided to take these original data sets without modification if it had fewer than 1000 samples. If the data set had more than 1000 samples, we downsampled the outlier ratio to 0.5% and use the derived version of the data set in our evaluation. As a result, we collected 45 derived outlier detection data sets for our performance evaluation and comparison.

We used the median AUC ROC as the performance metric to compare our methods against baselines as follows:

1. Median AUC ROC: For **Oracle** and **Single-best** baselines, we calculated AUC ROC scores using ground-truth labels for each data set. For each of these two base-

[5]https://github.com/t-autood/t-autood

[6]https://www.openml.org

[7]https://pypi.org/project/metaod/

[8]http://odds.cs.stonybrook.edu/

[9]https://www.dbs.ifi.lmu.de/research/outlier-evaluation/DAMI/

Metric	Single-best	Oracle	MetaOD	T-AutoOD
Median AUC ROC	0.637	0.785	0.559	0.669

Table 2: Summary performance on 104 internal benchmarks (at most 1% of samples are in minority class).

Metric	Single-best	Oracle	MetaOD	T-AutoOD
Median AUC ROC	0.829	0.976	0.785	0.900

Table 3: Summary performance on 45 external benchmarks (at most 0.5% of samples are in minority class).

lines, we obtained 104 AUC ROC scores for the internal benchmark and 45 AUC ROC scores for the external benchmark. Then, we took the median AUC ROC score from these scores. Therefore, **Single-best** and **Oracle** had two median AUC ROC scores, one for each benchmark.

2. Average Median AUC ROC: For **MetaOD** and **T-AutoOD**, we executed each algorithm for 10 runs, each with a different random seed. For each run, we obtained the median AUC ROC score across all data sets in each benchmark. This gave us 10 median AUC ROC scores. Finally, we took the average score across these 10 median AUC ROC scores and used it as the performance score of **MetaOD** and the **T-AutoOD** variations as shown in Table 2 and Table 3.

The following specific implementation of **T-AutoOD** was tested. The 20 most discriminative features were selected using the XGB approach [Chen and Guestrin, 2016]. Furthermore, we calculated the Mahalanobis distance of data instances in the data sets and added this distance as an additional feature. This was done in the data sets of both the transfer resource and the test collection. Therefore, the final representation contained a total of 21 features.

Table 2 shows the results for the internal benchmark. The main competitor **MetaOD** had an AUC of 0.559, which was significantly outperformed by **T-AutoOD** (AUC$= 0.669$) for a performance difference of 11%. The **T-AutoOD** also outperformed the **Single-best** approach. Surprisingly, **Single-best** also outperformed **MetaOD**, which is indicative of the fact that dataset-level features used by **MetaOD** might not have captured enough information in order to result in well-optimized performance. These results seem to resoundingly support the principle of using instance-level features. While the **Oracle** method performed the best, it is not implementable in a real-world setting where labels for the test data set are unavailable. Table 3 showed the performance comparison for the external benchmark. **T-AutoOD** outperformed **Single-best** by about 7% and **MetaOD** by 12%. Since the ROC AUC scores of the internal benchmarks were lower, it implies that these data sets were "more difficult" for outlier detection.

Since the total number of test data sets is too large to allow exhaustive presentation in detail, only the performance of **T-AutoOD** on external benchmarks will be provided in Table 4. These 45 data sets were chosen by varying size and dimensionality to show the robustness of **T-AutoOD** across data sets of different characteristics. The AUC performance of both variations of **T-AutoOD** as well as those of the baselines are shown for each data set in a row of the table. Amazingly, our methods outperformed the **Oracle** baseline for several data sets, such as **aloi (dami)**, **pima (dami)**, **stamps (dami)**, **spambase (dami)**, and **waveform (dami)**. This is possible since the performance scores of the **Oracle** baseline was selected using the best pipeline out of 400 pipelines while our methods combined the precomputed anomalous scores for its predictions. It needs to be pointed out that **MetaOD** did outperform **T-AutoOD** on some data sets, although this is to be expected for an unsupervised problem like outlier detection in which the results are evidentiary in nature.

Table 4 also includes the running time of **MetaOD** and **T-AutoOD**. For smaller data sets, **T-AutoOD** was faster than **MetaOD**, since **MetaOD** returns a string with estimator name and pairs of (name, value) of hyperparameters. String parsing is required to create an instance of the estimator before training and inference, which takes longer than using the **T-AutoOD**'s already instantiated pipelines to generate anomalous scores and perform inference with the meta-learner. However, the instance-specific nature of **T-AutoOD** can make it somewhat more expensive for larger data sets. For data sets with more than 5000 rows or more than 50 columns, **MetaOD** runs somewhat faster. However, since the running time is in seconds, the difference is not significant in practice and is worth the improvements in ROC AUC scores.

4 Conclusions and Summary

This paper presents a metalearning method for outlier detection with the use of a labeled transfer resource. All data sets in the transfer resource are converted into a uniform representation of anomaly scores. These scores are then combined using a meta-model that can be learned from the transfer resource because of the availability of ground-truth labels. Given a new data set (without labels), it can be transformed to the same score-based representation, and the learned model on the transfer resource can be used to classify points as anomalies. The use of a combination function of different pipelines makes this work a form of metalearning ensemble. Therefore, the work combines ideas from metalearning, transfer learning, and outlier ensembles. Experimental results with **T-AutoOD** show impressive performance on a wide variety of data sets. In future work, we will examine how to speed up **T-AutoOD** by leveraging the intuition that some outliers are much easier to identify than others — in such cases, the pipelines may need to executed only on the "difficult" instances of the data set.

Data set	**Oracle**	**SB**	**MOD**	**TOD**	**# rows**	**# cols**	**% OUT**	**MOD RT**	**TOD RT**
aloi (dami)	0.73	0.55	0.76	**0.85***	48267	27	3.04	26.82	309.94
annthyroid (odds)	0.99	0.92	0.91	**0.96**	6699	6	7.42	5.86	8.64
annthyroid (dami)	0.99	0.76	0.68	**0.90**	6628	21	5.0	4.31	16.28
arrhythmia (odds)	0.86	0.66	**0.73**	0.69	391	274	14.6	9.41	274.1
arrhythmia (dami)	0.86	0.65	**0.68**	0.6	249	259	20.0	8.14	231.8
breastw (odds)	1.0	**0.99**	**0.99**	**0.99**	449	9	34.99	3.29	0.88
cardio (odds)	0.98	0.86	0.87	**0.92**	1663	21	9.61	3.85	8.14
cardiotocogr. (dami)	0.96	0.79	0.83	**0.96***	1656	21	20.0	3.79	7.99
glass (odds)	0.92	**0.83**	0.47	0.73	210	9	4.21	3.34	0.79
glass_norm (dami)	0.93	**0.89**	0.87	0.75	210	7	4.21	3.5	0.71
heartdisease (dami)	0.97	0.8	**0.9**	0.69	155	13	19.79	3.62	0.84
hepatitis (dami)	0.87	0.62	**0.67**	0.63	72	19	9.46	3.81	0.88
ionosphere (odds)	1.0	**0.95**	0.91	0.62	230	33	35.9	3.48	1.2
ionosphere (dami)	0.98	**0.95**	0.89	0.63	230	32	35.9	3.42	1.17
letter (odds)	0.98	0.78	0.88	**0.95**	1507	32	6.25	6.53	9.87
lympho (odds)	1.0	0.95	0.97	**1.0***	147	18	4.05	3.41	0.92
lymphography (dami)	1.0	**1.0**	0.94	0.93	147	47	4.05	3.63	3.63
mammography (odds)	0.88	0.82	**0.83**	0.79	10977	6	2.33	4.57	18.42
mnist (odds)	0.98	**0.93**	0.92	0.78	6937	100	9.21	27.01	94.03
musk (odds)	1.0	**1.0**	**1.0**	0.94	2979	166	3.17	6.56	145.46
optdigits (odds)	0.99	0.84	0.81	**0.89**	5091	64	2.88	11.05	37.26
outlier_ecoli_norm	0.94	0.81	0.86	**0.93**	332	7	2.68	3.27	0.78
outlier_yeast	0.83	0.76	0.76	**0.82**	1305	8	4.76	3.71	3.82
ozone_onehr_outlier	0.71	0.38	0.4	**0.48**	2475	72	2.88	4.5	26.55
pageblocks (dami)	0.98	**0.95**	0.71	0.93	4907	10	4.98	3.9	10.45
parkinson (dami)	1.0	**0.99**	0.77	0.87	53	22	20.0	3.33	0.88
pendigits (odds)	1.0	**0.98**	0.76	0.81	6747	16	2.27	6.58	15.89
pima (odds)	0.82	**0.65**	0.57	0.52	505	8	34.9	3.82	2.08
pima (dami)	0.85	0.79	0.72	**0.89***	505	8	20.0	4.11	2.03
satellite (odds)	0.82	**0.81**	0.66	0.66	4421	36	31.46	5.45	13.56
satimage-2 (odds)	1.0	**1.0**	0.97	0.94	5760	36	1.22	4.28	15.35
shuttle (dami)	1.0	0.71	0.63	**0.98**	1005	9	1.28	3.48	1.72
smtp (odds)	0.97	**0.93**	0.72	0.91	6030	3	0.03	4.26	4.8
spambase (dami)	0.86	0.73	0.7	**0.88***	2540	57	20.0	4.32	26.73
stamps (dami)	0.96	0.94	0.92	**0.98***	314	9	4.92	3.64	0.83
thyroid (odds)	1.0	**0.99**	**0.99**	**0.99**	3697	6	2.47	4.87	5.37
vertebral (odds)	0.91	0.45	0.5	**0.65**	215	6	12.5	3.33	0.69
vowels (odds)	1.0	0.95	0.5	**0.97**	1413	12	3.43	3.7	4.84
waveform (dami)	0.9	0.61	0.59	**0.98***	3359	21	2.9	5.09	13.16
wbc (odds)	0.98	0.94	0.95	**0.97**	362	30	5.56	3.93	4.41
wbc (dami)	1.0	0.99	**1.0**	**1.0***	218	9	4.48	3.35	0.84
wdbc (dami)	1.0	**0.98**	0.5	**0.98**	362	30	2.72	3.64	5.19
wilt (dami)	0.95	0.42	**0.85**	0.78	4584	5	2.0	5.46	8.25
wine (odds)	1.0	0.72	0.5	**0.97**	124	13	7.75	3.27	0.86
wpbc (dami)	0.87	0.5	0.49	**0.53**	156	33	23.74	3.61	1.14

Table 4: This Table shows the performance on 45 data sets taken from ODDS and DAMI with different methods. The best performance in terms of the AUC score among **SL** (or **Single-best**), **MOD** (or **MetaOD**), and **TOD** (or **T-AutoOD**) is shown in bold. **% OUT** stands for "percentage of outliers" in the data set. The use of an asterisk demarcates when **T-AutoOD** outperforms or matches **Oracle**. For **MetaOD** and **T-AutoOD**, the numbers are Average AUC ROC across 10 runs with different random seeds. **T-AutoOD** wins 27 data sets while **MetaOD** wins 14 data sets, they tie for four data sets. **MOD RT** stands for **MetaOD Run time** and **TOD RT** stands for **T-AutoOD Run time**, both in seconds.

References

[Aggarwal and Sathe, 2015] Charu Aggarwal and Saket Sathe. Theoretical foundations and algorithms for outlier ensembles. *ACM SIGKDD Explorations Newsletter*, 17(1):24–47, 2015.

[Aggarwal and Sathe, 2017] Charu Aggarwal and Saket Sathe. *Outlier Ensembles*. Springer, 2017.

[Aggarwal, 2013] Charu Aggarwal. Outlier ensembles: position paper. *ACM SIGKDD Explorations Newsletter*, 14(2):49–58, 2013.

[Aggarwal, 2017] Charu Aggarwal. *Outlier Analysis*. Springer, 2017.

[Bahri *et al.*, 2022] Maroua Bahri, Flavia Salutari, Andrian Putina, and Mauro Sozio. Automl: state of the art with a focus on anomaly detection, challenges, and research directions. *International Journal of Data Science and Analytics*, 14(2):113–126, 2022.

[Bouneffouf *et al.*, 2020] Djallel Bouneffouf, Charu Aggarwal, Thanh Hoang, Udayan Khurana, Horst Samulowitz, Beat Buesser, Sijia Liu, Tejaswini Pedapati, Parikshit Ram, Ambrish Rawat, et al. Survey on automated end-to-end data science. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020.

[Burnaev *et al.*, 2015] Evgeny Burnaev, Pavel Erofeev, and Dmitry Smolyakov. Model selection for anomaly detection. In *Eighth International Conference on Machine Vision (ICMV 2015)*, volume 9875, pages 445–450. SPIE, 2015.

[Chandola *et al.*, 2009] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):1–58, 2009.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *ACM KDD Conference*, pages 785–794, 2016.

[Fan *et al.*, 2019] Xinjie Fan, Yuguang Yue, Purnamrita Sarkar, and YX Wang. A unified framework for tuning hyperparameters in clustering problems. *arXiv preprint arXiv:1910.08018*, 2019.

[Feurer *et al.*, 2015] Matthias Feurer, Jost Springenberg, and Frank Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[Franceschi *et al.*, 2017] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning*, pages 1165–1173. PMLR, 2017.

[He *et al.*, 2021] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-based systems*, 212:106622, 2021.

[Heller and Ghahramani, 2005] Katherine Heller and Zoubin Ghahramani. Bayesian hierarchical clustering. In *International Conference on Machine learning*, pages 297–304, 2005.

[Hutter *et al.*, 2011] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

[Lai *et al.*, 2021] Kwei-Herng Lai, Daochen Zha, Guanchu Wang, Junjie Xu, Yue Zhao, Devesh Kumar, Yile Chen, Purav Zumkhawaka, Minyang Wan, Diego Martinez, et al. Tods: An automated time series outlier detection system. In *AAAI Conference on Artificial Intelligence*, volume 35, pages 16060–16062, 2021.

[Lee *et al.*, 2019] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. Melu: Meta-learned user preference estimator for cold-start recommendation. In *ACM KDD Conference*, pages 1073–1082, 2019.

[Li *et al.*, 2008] Lihong Li, Michael Littman, and Thomas Walsh. Knows what it knows: a framework for self-aware learning. In *International Conference on Machine Learning*, pages 568–575, 2008.

[Li *et al.*, 2018] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

[Li *et al.*, 2020] Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. Copod: copula-based outlier detection. In *International Conference on Data Mining (ICDM)*, pages 1118–1123, 2020.

[Liuliakov *et al.*, 2023] Aleksei Liuliakov, Luca Hermes, and Barbara Hammer. Automl technologies for the identification of sparse classification and outlier detection models. *Applied Soft Computing*, 133:109942, 2023.

[Micenková *et al.*, 2014] Barbora Micenková, Brian McWilliams, and Ira Assent. Learning outlier ensembles: The best of both worlds–supervised and unsupervised. In *Outlier Detection and Description Workshop*, pages 51–54, 2014.

[Mourer *et al.*, 2023] Alex Mourer, Florent Forest, Mustapha Lebbah, Hanane Azzag, and Jerome Lacaille. Selecting the number of clusters k with a stability trade-off: an internal validation criterion. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 210–222. Springer, 2023.

[Pan and Yang, 2009] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[Real *et al.*, 2020] Esteban Real, Chen Liang, David So, and Quoc Le. Automl-zero: Evolving machine learning algorithms from scratch. In *International conference on machine learning*, pages 8007–8019. PMLR, 2020.

[Sathe and Aggarwal, 2016] Saket Sathe and Charu Aggarwal. Subspace outlier detection in linear time with randomized hashing. In *International Conference on Data Mining (ICDM)*, pages 459–468, 2016.

[Singh and Vanschoren, 2022] Prabhant Singh and Joaquin Vanschoren. Meta-learning for unsupervised outlier detection with optimal transport. *arXiv preprint arXiv:2211.00372*, 2022.

[Vaithyanathan and Dom, 1999] Shivakumar Vaithyanathan and Byron Dom. Generalized model selection for unsupervised learning in high dimensions. *Advances in neural information processing systems*, 12, 1999.

[Vartak *et al.*, 2017] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. A meta-learning perspective on cold-start recommendations for items. *Advances in neural information processing systems*, 30, 2017.

[Wistuba *et al.*, 2015] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Learning hyperparameter optimization initializations. In *International conference on data science and advanced analytics (DSAA)*, pages 1–10, 2015.

[Yao *et al.*, 2018] Quanming Yao, Mengshuo Wang, Yuqiang Chen, Wenyuan Dai, Yu-Feng Li, Wei-Wei Tu, Qiang Yang, and Yang Yu. Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306*, 2018.

[Zhao *et al.*, 2021] Yue Zhao, Ryan Rossi, and Leman Akoglu. Automatic unsupervised outlier model selection. *Advances in Neural Information Processing Systems*, 34:4489–4502, 2021.