JGR | Machine Learning and Computation

RESEARCH ARTICLE
10.1029/2024JH000192

Iterative Error-Driven Ensemble Labeling (IEDEL) Algorithm for Enhanced Data Quality Control for the Atmospheric Radiation Measurement (ARM) Program User Facility

Lishan Li[1], Kenneth E. Kehoe[1], Jiaxi Hu[1,2], Randy A. Peppler[1], Alyssa J. Sockol[1], and Corey A. Godine[1]

[1]Cooperative Institute for Severe and High-Impact Weather Research and Operations, University of Oklahoma, Norman, OK, USA, [2]NOAA/OAR National Severe Storms Laboratory, Norman, OK, USA

Key Points:

- Unsupervised learning methods can't generalize well to new data due to their reliance on the estimate of training data's anomaly ratio
- The Iterative Error-Driven Ensemble Labeling (IEDEL) algorithm effectively guides abnormal data pattern discovery in large data sets using pre-trained models
- The IEDEL algorithm reduces review effort by up to 95% without sacrificing accuracy in labeling abnormal data patterns

Supporting Information:

Supporting Information may be found in the online version of this article.

Correspondence to:

L. Li,
miali@ou.edu

Citation:

Li, L., Kehoe, K. E., Hu, J., Peppler, R. A., Sockol, A. J., & Godine, C. A. (2024). Iterative error-driven ensemble labeling (IEDEL) algorithm for enhanced data quality control for the atmospheric radiation measurement (ARM) program user facility. *Journal of Geophysical Research: Machine Learning and Computation*, *1*, e2024JH000192. https://doi.org/10.1029/2024JH000192

Received 8 MAR 2024
Accepted 19 JUN 2024
Corrected 28 SEP 2024

This article was corrected on 28 SEP 2024. See the end of the full text for details.

Abstract For over three decades, the Atmospheric Radiation Measurement (ARM) Program user facility has provided researchers with invaluable benchmark atmospheric data. Ensuring the accuracy and integrity of ARM data is vital, and to achieve this, the ARM Data Quality Office (DQO) has implemented customized quality control tests tailored to each variable, with guidance from instrument mentors. These tests are designed to pinpoint common issues, such as data exceeding valid ranges or persisting with little change over extended periods, and ARM offers tools for users to review and exclude contaminated data efficiently. However, certain quality issues, such as spikes in time series or data drift over time, sometimes evade detection by existing tests and require manual identification by data analysts and instrument mentors through visualization tools. To tackle these challenges more efficiently, the DQO has developed and implemented the Iterative Error-Driven Ensemble Labeling (IEDEL) algorithm with unanimous voting and transfer learning techniques to efficiently generate labeled data at scale. This initiative has empowered the creation of high-performing machine learning models, enabling real-time monitoring of data quality issues within the ARM data and thereby enhancing data integrity and accessibility.

Plain Language Summary For more than 30 years, the Atmospheric Radiation Measurement (ARM) Program user facility has been providing scientists with important atmospheric data. Ensuring these data are accurate and trustworthy is crucial. To achieve this, the ARM Data Quality Office (DQO) establishes tailored quality control (QC) checks for each data variable, based on thresholds designed by the ARM instrument mentors, who are experts in meteorology. These checks help identify common data issues, such as data falling outside the normal range or not changing as expected over time. However, some problems, like sporadic data spikes or shifts in the average of data over time, might not be detected by these QC checks. These issues require visual identification by data analysts and ARM instrument mentors using ARM's visualization tools. To become more efficient at detecting these problems, the DQO has developed a new method called the Iterative Error-Driven Ensemble Labeling algorithm to label data issues and used a machine learning algorithm to categorize them. This innovative approach enables the DQO to build intelligent applications that monitor data in real time, around the clock, and allow instrument mentors to resolve data issues promptly.

1. Introduction

The Atmospheric Radiation Measurement (ARM) Program has long maintained a dedication to collecting high-quality data for use by climate science researchers (see Turner and Ellingson (2016) for an overview of the program). Data quality processing has evolved over the life of the program (e.g., Peppler et al., 2016). Initially it was the responsibility of respective instrument mentors and site scientist teams, with early efforts focused on the development of self-consistency checks for individual datastreams and datastream intercomparisons. Data quality efforts were consolidated with the creation of the Data Quality Office (DQO) in 2000 to standardize data quality processing and analysis across the program and to provide consistent direction. The DQO today is responsible for monitoring and ensuring the integrity of all data collected across various ARM sites, which are equipped with a wide range of instruments. The DQO collaborates closely with ARM Infrastructure to continuously monitor and enhance the accuracy of data issue detection.

Specific quality control (QC) tests employed by the DQO have been designed and built to suit the unique requirements of each variable. These tests also help to identify time series data with erroneous values, such as those falling outside of a reasonable scientific range or exhibiting significant discrepancies between neighboring facilities. The thresholds of these QC tests are determined at the discretion of instrument mentors and others in the program. In addition, these QC tests are scrutinized by a group of data quality analysts who conduct further visual reviews to discover, validate, categorize, and report data issues within the ARM system.

To further tackle and automate time-series data anomaly detection tasks, the widely used approaches include: (a) Probabilistic-based methods, which involve training a model using archived data, and then using new data as input for the model's fitting test. If the fitting test fails, the new data is labeled as an anomaly (Markou & Singh, 2003). (b) Pattern matching-based approaches, which require known characteristics and supervision (Yu et al., 2015). (c) Clustering/distance-based methods, which rely on new input data having a lower relative distance to the archived data, classifying it as a normal sample (Chandola et al., 2009). (d) Predictive models, which use a regression-based method to label new data as anomaly points when the predictive data value significantly differs from the observed new data value (Giannoni et al., 2018). (e) Ensemble methods, which combine the aforementioned approaches and employ a voting mechanism to determine anomaly points (Iliopoulos et al., 2023). (f) A variant of semi-supervised learning (SSL), known as uncertainty-aware pseudo-label selection (UPS), which aims to enhance SSL performance by reducing label noise through the utilization of confidence probabilities in predictions (Nayeem Rizve et al., 2021). (g) Iterative Pseudo-Labeling algorithm, designed for Speech Recognition, which involves the iterative generation of pseudo labels, leveraging a small, labeled data set and a model trained with these data. This method requires no validation on label (Xu et al., 2020).

Leveraging machine learning (ML) for data QC offers significant advancements in efficiency and accuracy, reducing the need for manual monitoring efforts. This innovative implementation enables immediate alerts and timely interventions on data issues, thereby minimizing extensive data loss. While the benefits of ML are evident, the associated challenges are intricate and difficult to overcome.

Despite DQO's concerted efforts to implement QC tests for data quality assurance, certain types of data issues, such as spikes and drifts, occasionally evade QC scrutiny. Therefore, we require additional tools to identify these specific pattern-based data anomalies. Since the current data issue reporting system in ARM is not designed to seamlessly integrate with ML algorithms, one of the foremost challenges we must tackle is the lack of labeled data.

To initiate our ML project, we were first presented with two options. The first option involves dedicating an enormous amount of time to manually search for rare occurrences of data issues and label them until we accumulate enough labeled data. The second is to use unsupervised learning algorithms, which requires no labeling at all. Given the arduous nature of manually scrutinizing years of data to uncover rare anomalies, and considering that reducing the need for manual review was a primary motivation for launching our ML project, we have decided to start our project with the latter option. However, we later discovered that the results produced by unsupervised learning algorithms were not reliable. As a result, we conducted further investigation and developed innovative approaches to efficiently generate labels for building supervised learning models.

In this study, our primary focus was to develop a series of algorithms aimed at facilitating the implementation of SSL and transfer learning techniques for anomaly detection within the ARM Data Archive. The objective of anomaly detection in time series data is to pinpoint patterns or data points that exhibit significant deviations from the anticipated or normal behavior within a temporal sequence. To assess the effectiveness of our approaches, we conducted three distinct experiments. These experiments involved the utilization of 9-year relative humidity data, collected from 1 January 2013 to 31 December 2021, at the ARM North Slope of Alaska (NSA) site (e.g., Verlinde et al., 2016), and 13-year relative humidity and temperature data, collected from 28 September 2011 to 29 September 2023, at the ARM Southern Great Plains (SGP) site (e.g., Sisterson et al., 2016). We provide a comprehensive account of the performance metrics and outcomes associated with each of the proposed algorithms and models in this manuscript.

2. Methods

The initial motive for using ML algorithms in the ARM Data Archive was to enhance the efficiency and precision of data issue reporting. Building supervised learning models for identifying data issues is impossible without fully

labeled data. That's why we began our spike detection project with the exploration of unsupervised learning algorithms. We later discovered that certain unsupervised learning algorithms only yield satisfactory results when we accurately estimate the number of anomalies in the training data set. However, this number can vary significantly depending on the selected training data. For example, a single day might have multiple spikes, whereas an entire month might have none.

Consequently, we decided to build supervised learning models, which have more reliable performance than unsupervised learning models. To avoid the exhaustive task of manually labeling anomalies in the ARM data, we developed multiple algorithms for efficiently labeling anomalies in large-scale data sets. These algorithms enable us to train accurate supervised learning models, which will assist in real-time monitoring of data issues within the ARM Archive.

2.1. Unsupervised Learning

In the absence of labeled data, we delved into several prevalent unsupervised learning algorithms tailored for anomaly detection. These algorithms include Isolation Forest (Liu et al., 2008), One-Class Support Vector Machine (One-Class SVM) (Schölkopf et al., 2001), Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Schubert et al., 2017), and Local Outlier Factor (LOF) (Breunig et al., 2000).

The Isolation Forest algorithm constructs a random binary tree to isolate each training sample based on the premise that anomalies typically require more binary tree splits for isolation, whereas regular samples require much fewer. Its accuracy is considerably influenced by the "contamination" parameter, which denotes the expected proportion of anomalies in the data set. Calculating a contamination ratio is impossible without labeled data since it relies on the statistical analysis of historical data. Overestimating or underestimating data anomalies can lead to a surge of false positives (FPs) or False Negatives (FNs), respectively. In this context, "positive" refers to a sample containing at least one data spike, while "negative" denotes samples without any data spikes.

In contrast to the conventional SVM, which is widely used in classification, the One-Class SVM is designed to distinguish novelties from a data set by computing the maximum margin hyperplane that separates the training data from the origin. As a result, samples falling beneath this hyperplane are deemed anomalies.

DBSCAN, a density-based clustering approach, classifies samples residing in low-density regions as anomalies. Similarly, LOF detects anomalies by evaluating the local density deviation of a sample with respect to its neighbors. A significant local density deviation indicates a higher likelihood of the sample being an anomaly.

We evaluated the effectiveness of these four algorithms in identifying spikes within a time series using 10 days' worth of relative humidity data from the ARM datastream "nsametC1.b1," dated from 17 October 2020 to 26 October 2020. As illustrated in Figure S1 in Supporting Information S1, it's evident that the performance of the Isolation Forest is predominantly influenced by the selected "contamination" ratio, with other parameters set at their default values. In contrast, modifying the "n_estimators" parameter without changing the default "contamination" ratio showed no enhancement in accuracy. Although LOF also requires a contamination ratio for its model training, its default parameter values appear to be effective as displayed in Figure S2 in Supporting Information S1. Moreover, fine-tuning the "n_neighbors" parameter alone can yield rather accurate outcomes. DBSCAN, on the other hand, can achieve a similar precision by adjusting the "eps" and "min_samples" parameters, as shown in Figure S3 in Supporting Information S1. Amongst the four, the One-Class SVM was the least effective, outputting an excessively large number of FPs. More details are provided in Figure S4 in Supporting Information S1.

In summary, the primary obstacle with these algorithms lies in the selection of optimal values of model parameters. Without guidance from labels, model performance can be unstable and unpredictable, producing somewhat arbitrary predictions. As such, unsupervised learning algorithms might be better suited for data labeling tasks rather than for real-time anomaly detection in streaming ARM data.

2.2. Synthetic Method of Semi-Supervised Learning (SSL) and Transfer Learning

2.2.1. Semi-Supervised Learning

SSL has emerged as an effective strategy for building ML models, when there is a limited supply of labeled data and labeling additional data is prohibitively expensive. One common method within SSL involves using

clustering algorithms to group a mix of labeled and unlabeled data into several classes. After that, labels of labeled data are propagated to the unlabeled data within each cluster. Additionally, various other methods in SSL assign pseudo-labels based on prediction probabilities. However, these methods offer no control over the accuracy of their labels. The high level of uncertainty and noise associated with these labels often results in creating less robust supervised learning models. To overcome this challenge, we propose an innovative and practical approach that enhances labeling efficiency without sacrificing accuracy. Our approach begins with the concept of transfer learning.

2.2.2. Transfer Learning

Transfer Learning is a generic term in ML, referring to a technique of using a pre-trained model designed and built for a problem to solve another new problem (Божиновски, 2020). It assumes that the knowledge gained in learning a previous task can be insightful in generalizing one another. When a suitably labeled data set for transfer is accessible, transfer learning could be a more favorable option over any SSL techniques (Oliver et al., 2018). Considering the enormous number of variables across a wide variety of instruments at different ARM sites, building one model for each time series might seem overly ambitious and highly inefficient. As a result, it's inevitable for us to leverage transfer learning techniques to reduce our review workload. To determine how to make it work and whether it's an appropriate approach, we will expand our discussion with results of three experiments from our spike detection project.

2.2.3. Iterative Error-Driven Labeling (IEDL)

There's a self-explanatory trade-off between the effort required by labeling and the accuracy of labels. The challenge lies in how to achieve a high level of label completeness of true positive samples with a minimal manual workload. Unlike the classic pseudo-labeling approach (Nayeem Rizve et al., 2021), where a pre-trained model filters predictions on unlabeled data solely based on one or multiple thresholds of the corresponding predictive probabilities, our proposed semi-supervised labeling method demands iterative review for positive predictions and prediction errors. The first algorithm we developed in this study, termed Iterative Error-Driven Labeling (IEDL), represents a synthesis of both SSL and transfer learning techniques, and is specifically tailored for labeling the minority class in large-scale data sets with boosted efficiency.

Labeling samples into both minority and dominant classes is time-consuming. Hence, IEDL focuses on differentiating and labeling samples of the minority class in contrast to the majority class samples. Initially, IEDL assumes that all unlabeled samples belong to the majority class and utilizes a pre-trained model with labeled data to make predictions on unlabeled data. It then iteratively updates labels for true positive samples by scrutinizing prediction errors until no more new errors are found. Since labels of the minority class samples are created after validation, instead of reviewing both FP and FN samples, we only need to review the FP ones. However, in scenarios where labeling criteria are uncertain or difficult to quantify—such as deciding whether an abnormal data pattern signifies a significant data quality issue that warrants further investigation, it's advisable to review both FP and FN samples.

To better explain the IEDL algorithm, let's consider a simple example with an illustration. Figure 1 shows a small, labeled mixture of samples with and without anomalies, with samples without anomalies being the dominant "negative" class. To identify all the anomaly samples without reviewing the entire unlabeled data set, we employ the IEDL algorithm. The process begins by training a model on this small set of labeled data, which is then used to make predictions on unlabeled data. This technique is known as transfer learning, notably reducing the effort of reviewing the whole data set to focusing solely on samples predicted as positive. However, not all positive predictions are accurate, since the effectiveness of transfer learning depends on the level of similarity between the previous task and the current one. Next, iterative reviews on prediction errors occur to further reveal the remaining true positive samples. In this example, by mainly reviewing regular samples that are falsely predicted as anomalies, we can incrementally locate all or at least most anomaly samples, ultimately reducing label noise and prediction uncertainty.

It is worth noting that, at the end of the IEDL implementation, samples initially marked with the pseudo label "negative" are only partially confirmed as either "positive" or "negative" after manual examination, as highlighted in red rectangles in Figure 1. When samples first assigned the pseudo label "negative" clearly don't match the target data pattern (in this project, the shape of a spike), it is highly probable that these samples remain

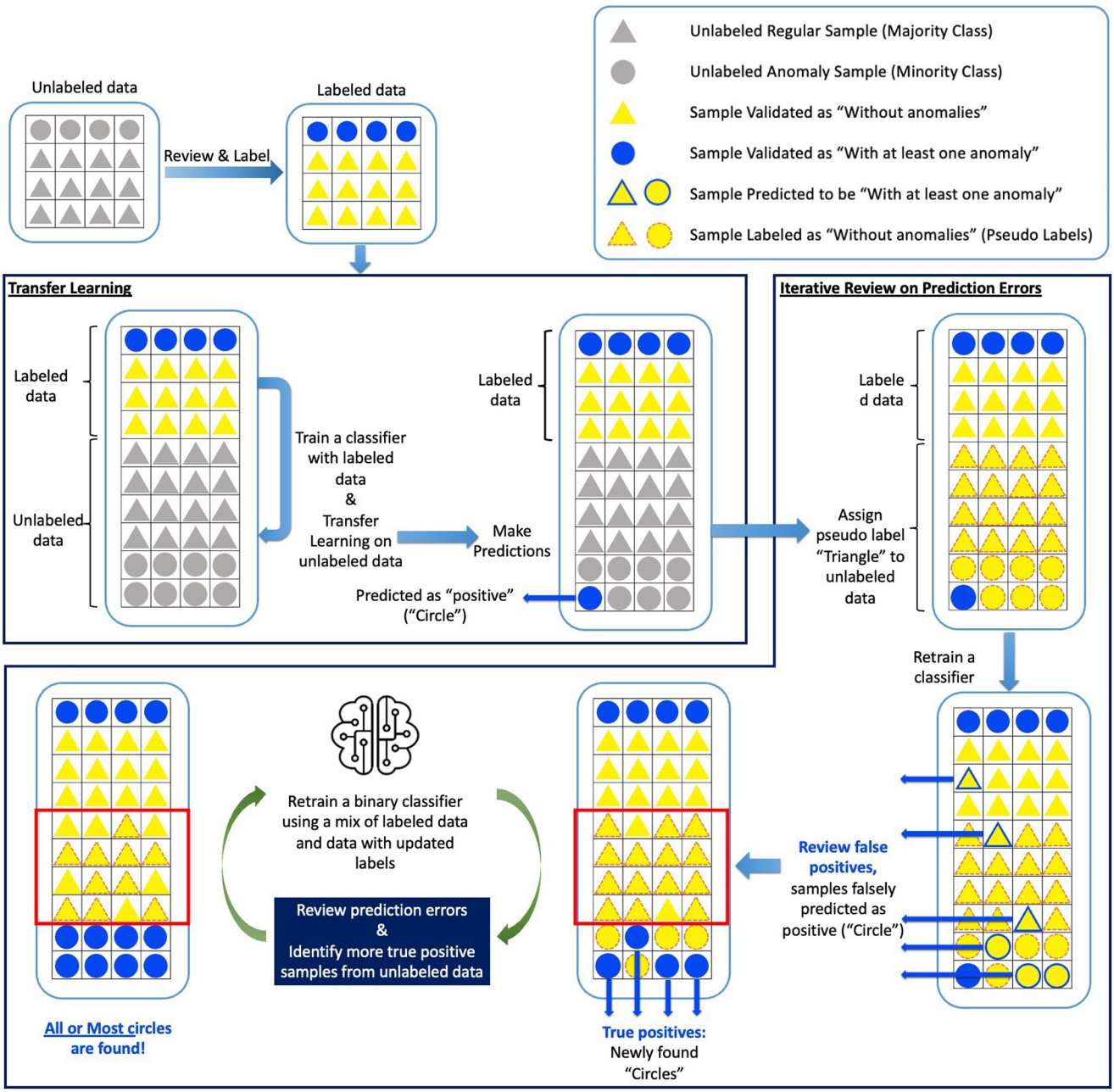

Figure 1. An illustrative process of Iterative Error-Driven Labeling (IEDL) algorithm. Shape and color legends of unlabeled, validated, and predicted data samples are shown in the upper right corner.

unreviewed due to the lack of ambiguity in their classification. They are most likely to be correctly labeled, as no new FP samples were revealed at the end, reflecting greatly minimized label noise. This is how IEDL enables us to label data quality issues without investigating the entire data set.

The concept of IEDL draws inspiration from the Gradient Descent (GD) Algorithm, predominantly used in Deep Learning. GD involves iterative updates to model parameters in the direction of minimizing the loss function. Instead of calculating gradient descents of the loss function, as illustrated in Figure 2, IEDL relies on iteratively examining prediction errors to guide the labeling process toward reducing the number of remaining unrevealed true positive samples. We also defined a threshold for the number of samples required for review in each iteration of IEDL, which we term as the "desired validation effort (DVE)," comparable to the learning rate in GD. A large

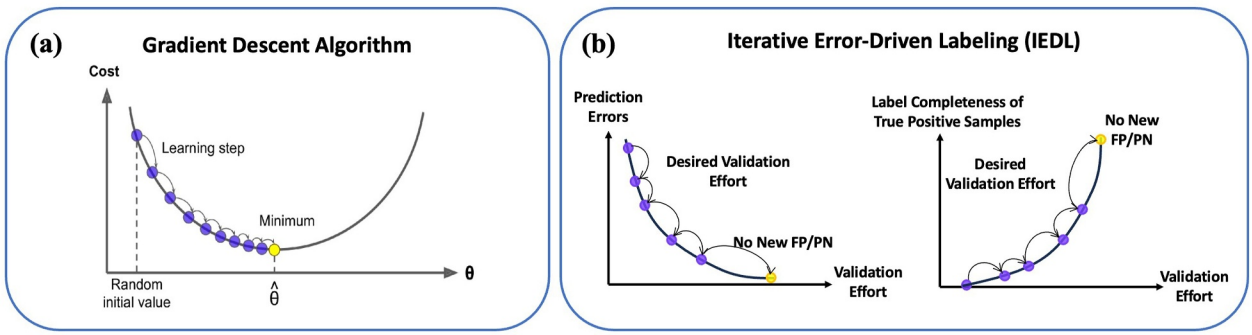

Figure 2. Comparison between Gradient Descent (GD) (a) and Iterative Error-Driven Labeling (b) algorithms.

learning rate can possibly lead to divergence in GD, but in IEDL, a large value of DVE will result in fewer iterations without deviation.

An overfitting model can learn the training data by memorizing each training sample, resulting in poor performance in generalization. To enable a model to discover unrevealed true positive samples from the unlabeled data set, we need to intentionally opt for an underfitting model, as proposed in Table 1.

There are several methods to tune down the model complexity, which are inspired by the concept of the ensemble learning algorithm, that builds a strong learner by aggregating results from a bunch of weak learners. These include: (a) reducing the number of training samples used in weak learner training; (b) selecting a small subset of available features for training a weak learner; and (c) choosing a less complex ML algorithm either by setting regularization constraints within model hyperparameters or by using a simpler algorithm by nature. By applying any combination of these techniques, we can effectively prevent the use of an overfitting model in IEDL.

2.2.4. Iterative Error-Driven Ensemble Labeling (IEDEL)

The Iterative Error-Driven Ensemble Labeling (IEDL) algorithm is a synthetic method of SSL and transfer learning that improves efficiency over the blind and manual searching and labeling anomalies in large-scale data sets. However, it has shortcomings from two perspectives. First, IEDL relies on predictions from merely one

Table 1
Iterative Error-Driven Labeling Algorithm Information for Anomaly Detection

Iterative error-driven labeling (IEDL) algorithm for anomaly detection
Data: Labeled data $D_L = \{(X_i, y_i)\}_{i=1}^{N_L}$, Unlabeled data $D_U = \{(X_i, y_i)\}_{i=1}^{N_U}$
Labels: $y_i =$ "positive," the minority class (with at least one anomaly). $y_i =$ "negative," the majority class (without any anomalies)
Process:
(Transfer Learning)
1. Assume all samples in D_U belong to the majority class, $\tilde{D}_u = \{X_i, y_i =$ "negative"$\}_{i=1}^{N_U}$
2. Train a non-overfitting classifier with labeled data D_L
3. Use this trained model to make predictions on unlabeled data D_U
4. Review samples predicted as positive and update labels to the minority class, "positive," in $\tilde{D}_u$ on newly discovered true positive samples, as $\tilde{D}_u'$
5. Merge D_L and $\tilde{D}_u'$ and marked it as D_L', so D_L' will be the labeled data under development
(Iterative Labeling)
Repeat
1. Retrain a non-overfitting classifier with D_L'
2. Prediction Error Review: Review prediction errors and record reviewed samples to avoid duplicate reviews in the following iterations
3. Update labels of D_L' with newly found true positive samples from this review
Until there are no prediction errors that haven't been reviewed yet

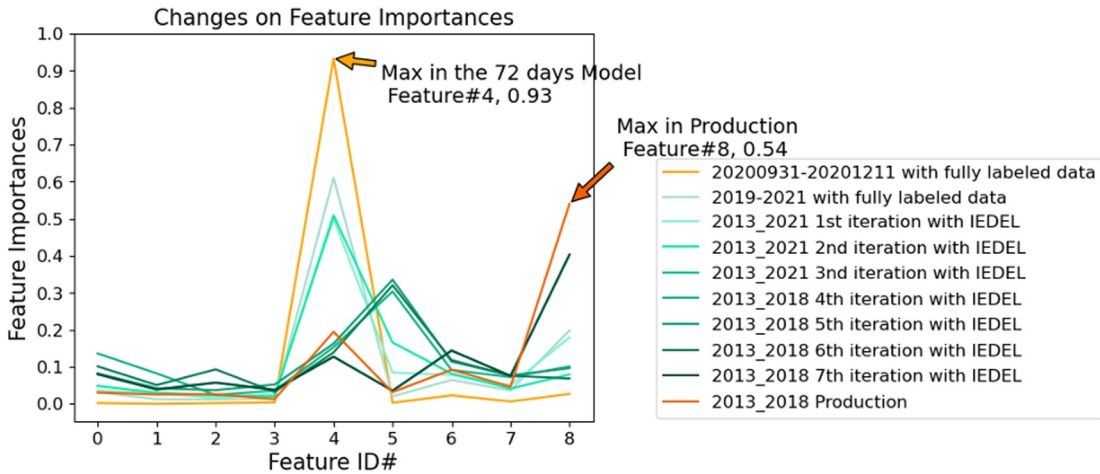

Figure 3. Changes on Feature Importance in Experiment I with nsametC1.rh_mean data. It compares the initial model, which was trained on 72 days of data, with the final model, trained on data spanning 6 years.

model; thus, the return on review effort, reflected by precision, is uncertain, which is later clearly demonstrated in Figure 7. There are hardly any measures to ensure high precision since the IEDL algorithm employs only a single model during transfer learning.

Furthermore, it's possible that the small set of manually labeled data used initially in the IEDL algorithm might not comprehensively represent a full spectrum of possibilities. Evidence of this can be seen in Figure 3, where the maximum feature importance of nine available features starts at 0.93 with 72 days of labeled data and drops to 0.54 when using 6 years of labeled data (from 2013 to 2018). This change indicates that the model trained on 6 years of data learned to utilize more features in classifying training samples, whereas the 72-day model relied heavily on one feature for predictions. Given this observation, pseudo labeling and its advanced variant, the uncertainty-aware pseudo-label selection framework (UPS), are not ideally suited for efficiently generating accurate labels for anomalies within the ARM Archive.

To address these shortcomings, instead of using a single underfitting model for labeling anomalies, we can leverage multiple underfitting models. These models are constructed by selecting only a subset of features during training, thereby forcing each model to focus on different aspects of the data during classification. By aggregating results from several weak learners, we traded off more bias for less variance, therefore reducing training loss. This novel approach is called the IEDEL algorithm, which is a variant and more powerful version of the IEDL algorithm.

In IEDEL, we first build a series of underfitting models with different subsets of features using labeled data and make predictions on unlabeled data with these models. After reviewing positive predictions on unlabeled data, we will generate more labels on the minority class. An underdeveloped set of labeled data will be utilized in iterative prediction error review after combining fully labeled and partially labeled data. Table 2 displays the IEDEL algorithm's process step by step with more details. Additionally, we suggest employing unanimous voting to effectively reduce the number of positive predictions requiring review in IEDEL.

Both the IEDL and IEDEL algorithms share a common weakness: without manually validating the entire data set, we cannot ensure 100% labeling accuracy for true positive samples. Nevertheless, these methods are highly efficient in locating most true positive samples that are sufficient to build robust models for identifying the targeted data issue in future data, thus fulfilling their intended purpose.

2.2.5. Unanimous Voting

In IEDEL, several underfitting models output predictions to aid in disguising unrevealed true positive samples. With either hard voting or soft voting methods to select samples predicted positive for review, there's no guarantee of a high return on effort spent in validation. To address such a dilemma, we propose a new algorithm, termed unanimous voting. This approach is detailed step-by-step in Table 3. Unanimous voting requires predictions from

Table 2
Iterative Error-Driven Ensemble Labeling (IEDEL) Algorithm Information for Anomaly Detection

Iterative error-driven ensemble labeling (IEDEL) for anomaly detection
Data: Labeled data $D_L = \{(X_i, y_i)\}_{i=1}^{N_L}$, Unlabeled data $D_U = \{(X_i, y_i)\}_{i=1}^{N_U}$
Labels: y_i = "positive," the minority class (with at least one anomaly). y_i = "negative," the majority class (without any anomalies)
Process:
(Transfer Learning)
1. Assume all samples in D_U belong to the majority class, $\tilde{D}_u = \{X_i, y_i = \text{"negative"}\}_{i=1}^{N_U}$
2. Train a series of underfitting classifiers with labeled data D_L
3. Use these trained models to make predictions on unlabeled data D_U
4. Aggregate results from these models by either hard voting, soft voting (with or without weights), or unanimous voting
5. Review samples predicted as positive from the aggregated results and update labels to the minority class, "positive," in $\tilde{D}_u$ on newly discovered true positive samples, as $\tilde{D}_u{}'$
6. Merge D_L and $\tilde{D}_u{}'$ and marked it as $D_L{}'$, so $D_L{}'$ will be the labeled data under development
(Iterative Labeling)
Repeat
1. Retrain a series of underfitting classifiers with $D_L{}'$
2. Prediction Error Review: Review prediction errors and record reviewed samples to avoid duplicate reviews in the following iterations
3. Update labels of $D_L{}'$ with newly found true positive samples from this review
Until there are no prediction errors that haven't been reviewed yet

all selected subset models to agree on voting a sample to the minority class to predict it as a minority class sample, which means we only review the minority class predictions with the highest aggregated votes.

In Figure 4, we compare unanimous voting with hard voting and soft voting. In scenarios where hard voting (Figure 4a) produces minority class predictions, unanimous voting (Figure 4c) is more likely to favor the majority class just because not all predictors voted for the minority class. In this illustration, we assume that the weights of weak learners are equal within the ensemble. With hard voting, the ensemble learning result is determined based on the majority vote. For instance, as shown in Figure 4a, a sample is predicted as "0" because class "0" receives the most votes. In contrast, soft voting (Figure 4b) calculates the arithmetic mean of the votes based on predictive probabilities. In Figure 4b, even though class "0" receives two votes, the sample is predicted as "1." This distinction can lead to situations where the results of hard voting diverge from those of soft voting. Unanimous voting, however, requires all weak learners to simultaneously vote for the minority class to predict that class, thereby making it more likely to classify samples as belonging to the majority class. As shown in Figure 4c, the sample is voted as "1" because one of the learners voted for "0."

While unanimous voting has proven effective in reducing the overall review effort, it can, unfortunately, result in producing no positive predictions when too many subset models are used. Therefore, we established a threshold for the DVE per iteration in our project. For example, we can set the DVE at 50 samples. We continuously

Table 3
Unanimous Voting for Ensemble Labeling Algorithm Information

Unanimous voting for ensemble labeling
Define:
Labels: y_i = "positive," the minority class (with at least one anomaly). y_i = "negative," the majority class (without any anomalies)
We observe voting results from N classifiers, $V = \{V_1, \ldots, V_i\}_{i=1}^{N}$
Process:
1. If any V_i = "negative," the result of unanimous voting is "negative"
2. Otherwise, the result of unanimous voting is "positive"

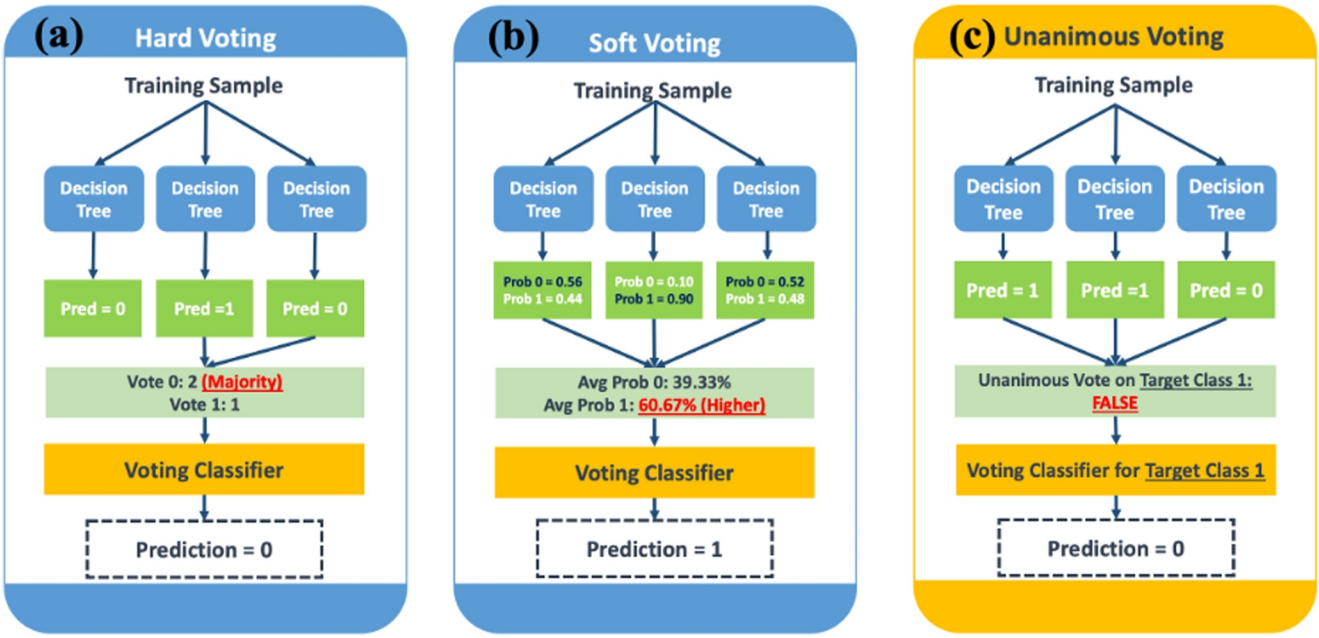

Figure 4. Comparison of Three Voting Algorithms in Ensemble Learning. (a) "Hard Voting" algorithm where the final prediction is the majority vote from multiple decision trees. (b) "Soft Voting," which takes the average probability from the predictions of each tree to determine the final prediction. (c) "Unanimous Voting," requiring all decision trees to agree on a positive prediction for it to be considered true, otherwise the final prediction defaults to negative. Each panel visualizes the flow from input through ensemble learning to the collective decision of the ensemble classifier. "1" denotes "positive," and "0" indicates "negative."

employed additional subset models in unanimous voting to generate more rigorous predictions until the number of samples falsely predicted as positive fell below 50. By doing so, we make sure to continuously review a certain amount of new prediction errors throughout iterations until no new ones are found, achieving a level of labeling completeness of the minority class, wherein the model can easily differentiate between two classes due to the substantial reduction in labeling noise.

3. ARM Data

3.1. ARM Data and Experiment Setting

ARM collects large amounts of continuous atmospheric data from over 200 meteorological instruments—roughly 50 terabytes of data per month (U.S. Department of Energy, 1998). These data come from the instruments as raw data, then flow through a number of processes to ensure accuracy. Raw data is ultimately transformed into useable netCDF files through the ingest process. It is during this process that various data levels are generated: a-level data, which converts the raw data into netCDF files; b-level data, which applies various types of QC checks depending on the instrument; and c-level data, which are considered Value Added Products and may include more complex derivations and QC. ARM data is also subject to stringent data standards to ensure better data quality, readability, and reliability. ARM datastreams in particular follow the following format (ARM Standards Committee, 2020):

(**sss**)(**inst**)(**facility**).(**level**)

sss is the three letter ARM site identifier
inst is the ARM instrument abbreviation
facility is the two or three character ARM facility designation
level is the data level descriptor consisting of one lowercase letter followed by one number

Our experiments are aimed to discover the best practice of efficient labeling and transfer learning of this ARM data from various time series data across different variations, addressing problems as follows:

Table 4
Data Sets Used in This Study, Including Atmospheric Radiation Measurement Site, Instrument, Facility, and Variable Information

Data sets	Site	Instrument	Facility	Variable
nsametC1.rh_mean	North Slope of Alaska (NSA)	MET	Central Facility 1 (C1)	Relative Humidity Mean (rh_mean)
sgpmetE37.rh_mean	Southern Great Plains (SGP)	MET	Extended Facility 37 (E37)	Relative Humidity Mean (rh_mean)
sgpmetE37.temp_mean	Southern Great Plains (SGP)	MET	Extended Facility 37 (E37)	Temperature Mean (temp_mean)

- Transfer learning across time
- Transfer learning across ARM sites
- Transfer learning across different variables

For these experiments, we will be using data from two ARM datastreams: nsametC1.b1 and sgpmetE37.b1 (Kyrouac et al., 2011; see Table 4). Keeping in mind the ARM datastream format mentioned above, it is clear that "NSA" (or the North Slope of Alaska) is the site and C1 is the facility for the nsametC1.b1 datastream, while "SGP" (or the Southern Great Plains) is the site and E37 is the facility for the sgpmetE37.b1 datastream. From the nsametC1.b1 datastream, we will use mean relative humidity (rh_mean) data. From the sgpmetE37.b1 datastream, we will use mean relative humidity (rh_mean) data as well as mean temperature (temp_mean) data. All of the data from these datastreams are collected from two of ARM's Surface Meteorology Systems (MET instruments), which use in-situ sensors to gather information. The data collected from both datastreams will be b-level data, meaning that there will be QC checks applied to many of the measurement variables, including both rh_mean and temp_mean.

3.2. Notation for Spike Detection Experiments

Let $D_L = \{(X_i, y_i)\}_{i=1}^{N_L}$ be a labeled data set with N_L samples, where X_i denotes the feature space and y_i represents the corresponding class label. The objective of this project is to detect abnormal data spikes that are significantly indicative of a data quality issue, warranting a review by the DQO. The decision to include a data spike in an ARM data quality report rests with both the DQO and the instrument mentors. ML models are used solely for monitoring purposes. There are two classes: "positive," for samples exhibiting at least one suspicious data quality spike, and "negative," for samples without such spikes. Given the rarity of data quality spikes, the "negative" class is the majority class.

Let $D_U = \{(X_i)\}_{i=1}^{N_U}$ be a data set with N_U samples but without any labels. In D_U, the majority class label, "negative," will by default be assigned to all samples, $\{(X_i, \tilde{y}_i = \text{"negative"})\}_{i=1}^{N_U}\}$, referred to as pseudo labels. By applying our proposed algorithms, we can incrementally update labels of true positive samples from "negative" to "positive" within these pseudo labels. This process eventually generates a set of labels for D_U with high accuracy, enabling us to construct high-performance supervised learning models to detect data quality spikes in the ARM Data Archive.

3.3. Data Preprocessing

Most ARM datastreams are time series data. To incorporate the time dimension in model training, we generated sliding windows based on a fixed time window size and stride, using them as training samples. In our spike detection project, the time window size is set to 20 timesteps (with a time resolution of 1 min), and the stride varies based on the need for up-sampling and down-sampling, which we will explain in Section 3.4.

Based on our experiments, we've established a rule of thumb that a desired time window size for anomaly detection must be large enough to provide sufficient information about changes over time to determine whether a window contains a targeted type of data anomaly. However, it should also be small enough to isolate this anomaly within a single time window. For instance, a time window size of one day would not be appropriate, as multiple spikes can occur within one day. When multiple anomalies occur within a single time window, they might no longer appear to be abnormal patterns in time series data, leading to compromised model performance. Furthermore, the larger the window size, the longer the wait to process and classify it. With all these in mind, there is a range of appropriate window sizes. As the DQO, we prefer to provide our users with more timely and precise detection of data issues; therefore, a smaller time window size within this range is better suited to our needs.

As we observed from data sets used in experiments, the likelihood of a 20-min sliding window containing more than one spike is very low. In scenarios where multiple spikes occur within a 20-min time window, our proposed solution remains effective since the successive sliding windows can always isolate the first and the last data spikes, provided the stride is set to 1. Thus, we suggest setting stride to 1 when implementing ML models for production to ensure proper handling of samples where multiple spikes appear in a single time window. Next, we will expand the discussion on how to select a proper stride for slicing sliding windows to address class imbalance issues in anomaly detection.

3.4. Class Imbalance Issue and Data Augmentation

Data quality issues in the ARM Archive are rare. When training a binary classifier to identify a data issue, it's crucial to first address the class imbalance issue. This prevents models from favoring the majority class with lazy predictions, instead of learning to differentiate classes effectively. Generally, we could consider up-sampling the minority class by introducing random noise into existing samples of this class and incorporating them into the training set. For example, the Synthetic Minority Over-sampling Technique (SMOTE) is useful for generating synthetic data points interpolated between existing minority class samples. However, SMOTE isn't suitable for identifying abnormal data patterns in time series data, as it does not guarantee the preservation of the overall pattern (Chawla et al., 2002).

Another method of up-sampling, specific for time series analysis, is to augment the number of the minority class samples by creating random time windows around that identified data anomaly. For example, a data spike in our spike detection project with 1-min resolution relative humidity data might span 3–5 min. Each sample represents a 20-min time window. We can label multiple time windows containing this data spike as "positive," rather than labeling just one sample per spike. This approach upscales the minority class set by setting the stride to its minimum value. Additionally, such an augmentation technique ensures that labels for spikes are location-irrelevant within a 20-min time window, enhancing the robustness of our spike detection models, especially when the number of strides varies in segmenting sliding time windows.

Alternatively, we could apply down-sampling to the majority class samples by randomly selecting a subset of a size comparable to the minority class set. Moreover, by setting the stride to a larger value, such as half of the time window size, when producing sliding windows for samples without spikes, we can effectively decrease the number of majority class samples. By implementing both up-sampling and down-sampling techniques, we have successfully mitigated the impact of extreme class imbalance in our anomaly detection project.

4. Results

In this section, we present the results of our proposed algorithms, beginning with a comparison between IEDL and IEDEL. This comparison demonstrates the advantages of using an ensemble of subset models for labeling anomalies in large-scale data sets, as opposed to relying on a single model. Additionally, it highlights the importance of the unanimous voting algorithm in effectively improving the return on review effort. Regarding ML algorithms, we employed the Decision Tree algorithm for data labeling, and subsequently, the Random Forest algorithm was used for processing tasks on the labeled data.

4.1. IEDL and IEDEL

As illustrated in Figure 5, the IEDEL algorithm gradually improved the precision of labeled data by iteratively increasing the completeness of true positive samples during reviews of FPs. Each row presents results for one data set in the following order: "nsametC1.rh_mean," "sgpmetE37.rh_mean," and "sgpmetE37.temp_mean." In Figure 5a, both FPs and FNs initially fluctuated over several iterations, vividly illustrating how label noise misled the models. However, they eventually stabilized at a point where no new FPs were identified in the last iteration. As of now, there remain some prediction errors, including both FPs and FNs, but all have been investigated in past iterations, indicating that label noise has been significantly reduced. This suggests the need to switch to a more complex model and refit the data. Based on our results from these experiments, we have found few new anomalies after refitting. Nonetheless, there is still a possibility of discovering new anomalies by reviewing prediction errors from the refitted models, but these should be minimal, and manually labeling them is no longer an exhaustive task. By employing the ARM visualization tools, including DQ-Plotbrowser and DQ-Zoom, which present data on a daily basis, we used the number of dates of reviewed data to indicate the workload in each iteration of IEDEL. In

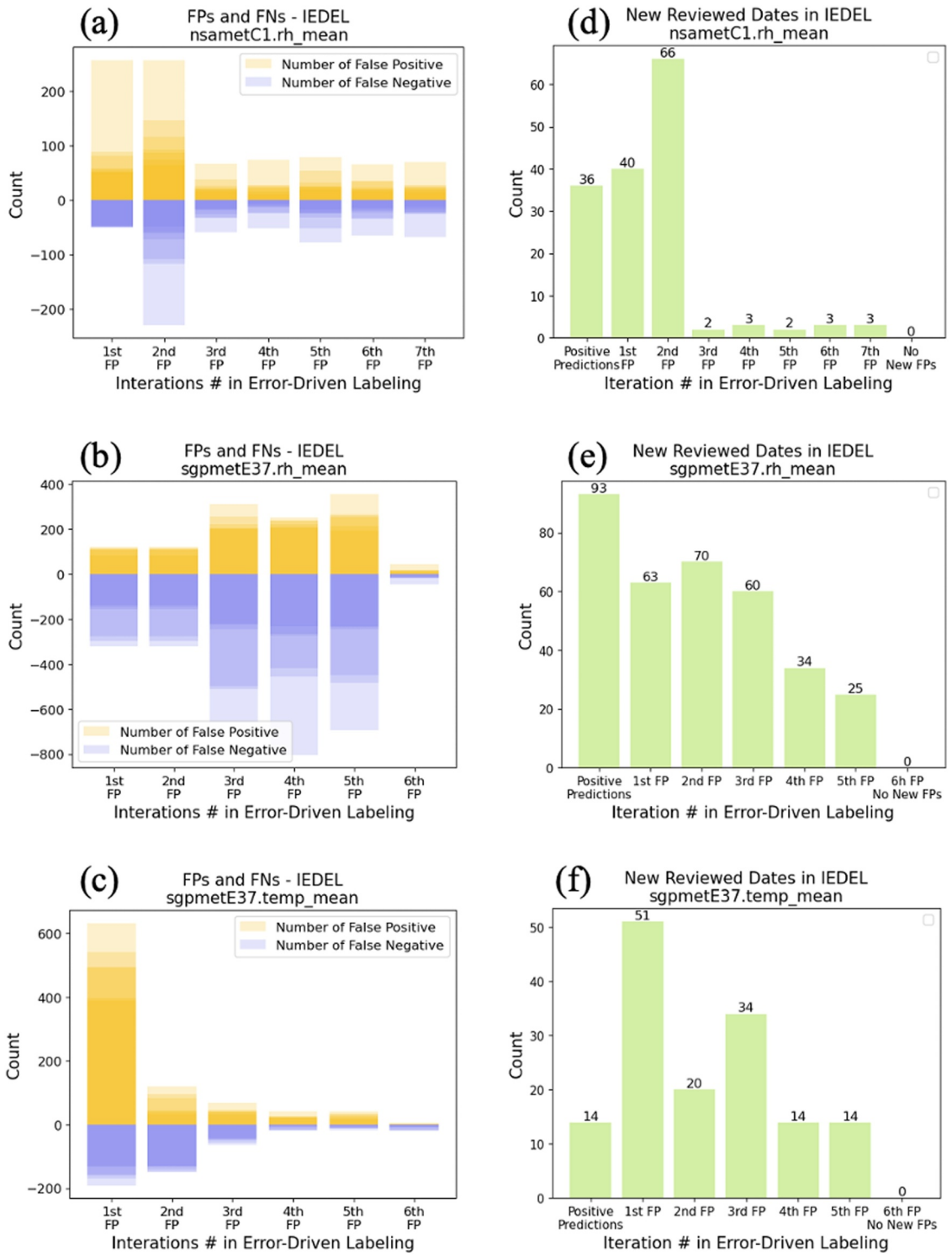

Figure 5. Analysis of Prediction Errors and Reduction of Review Efforts in Iterative Error-Driven Ensemble Labeling (IEDEL). Left column panels (a), (b), and (c) show the count of False Positives (FPs) and False Negatives (FNs) across successive iterations of error-driven labeling for the data sets "nsametC1.rh_mean," "sgpmetE37.rh_mean," and "sgpmetE37.temp_mean," respectively. Right column panels (d), (e), and (f) display the number of new reviewed dates across the same iterations for each data set. These results collectively exhibit the progression of the IEDEL algorithm.

Table 5
Summary of Data Review and Efficiency Gain Using the Iterative Error-Driven Ensemble Labeling Algorithm

Data sets	Total number of days	Number of reviewed days	Number of days with DQ spikes after review	Reduction in review effort (%)
nsametC1.rh_mean	3,287	155	40	95.3
sgpmetE37.rh_mean	4,385	345	96	92.1
sgpmetE37.temp_mean	4,385	147	23	96.8
Total	12,057	647	159	94.6

Note. This table presents the total number of dates, the number of dates reviewed, the number of dates with detected data spikes after the review, and the corresponding reduction in review effort for the data sets "nsametC1.rh_mean," "sgpmetE37.rh_mean," and "sgpmetE37.temp_mean." The aggregated totals for all data sets are also provided in the last row, demonstrating the overall performance of the IEDEL algorithm on enhancing review efficiency.

Figures 5b, 5c, 5e, and 5f, similar observations can be found in the data sets "sgpmetE37.rh_mean" and "sgpmetE37.temp_mean."

One thing worth noting is that the number of iterations required in IEDEL depends on the validation effort invested in each iteration. Therefore, such a number might differ across experiment settings. Also, we implemented unanimous voting until the number of aggregated positive predictions fell below the DVE. The values of DVE used in our spike detection project range from 50 to 100.

The purpose behind the IEDEL algorithm is to efficiently label anomalies within a large-scale database, such as the ARM Archive. By implementing IEDEL, we achieved a significant reduction in labeling efforts for our spike detection project, with reductions of 95.3%, 92.1%, and 96.8% for data sets "nsametC1.rh_mean," "sgpmetE37.rh_mean," and "sgpmetE37.temp_mean," respectively (shown in Table 5 and Panels a, b, and c of Figure 6). Notably, labels for 13 years of "sgpmetE37.rh_mean" data were derived through transfer learning from just 72 days of data from "nsametC1.rh_mean," resulting in only 39.3% of the reviewed samples being accurately identified as data spikes, as illustrated in Figure 6e. Furthermore, the application of transfer learning from "sgpmetE37.rh_mean" to "sgpmetE37.temp_mean" resulted in a mere 15.6% accuracy in spike detection among our reviewed samples. These compromised outcomes can be attributed to the different levels of similarity between the data sets mentioned. Despite these challenges, IEDEL significantly reduced our average review time by 94.6% across these data sets.

Instead of reviewing data points to locate rare anomalies, we now only need to review a minimal amount of data to identify most of the targeted data issues. For example, instead of reviewing 12 years of "sgpmetE37.rh_mean" data to locate rare occurrences of spikes, we now only need to review 1 year of data to identify most of the spikes. IEDEL can enhance labeling efficiency without compromising accuracy, empowering us to construct supervised models that offer precise and real-time monitoring for various data quality issues within the ARM Archive.

4.2. Unanimous Voting in IEDEL

Unanimous voting is a rigorous algorithm employed to make predictions that were with high precision and low recall from an ensemble of underfitting models. This method aligns well with our goal of anomaly detection while minimizing the validation effort. In Experiment I, we trained models with 72 days of data, which included 110 recurring spikes from 30 September 2020 to 11 December 2020. We then utilized these well-trained models to predict using data from 1 January 2019 to 31 December 2021, excluding the 72 days in 2020. When examining the number of positive predictions, we observed a range varying from 224 to 1,431 after merging overlapping time intervals. Reviewing 224 time intervals in one iteration is still less than ideal, given the scale of the ARM Archive, especially considering that later we found that only 22% of these 224 time intervals were correct. With unanimous voting, we were able to significantly reduce this range to only 41 positive predictions, which were later examined with a precision of 51%. This resulted in a significant enhancement in return on review effort.

We also observed that the number of positive predictions produced by ensemble models, trained using data from 2019 to 2021 and applied to data from 2013 to 2018, reduced from a range of 252 to 4,559 to a mere 42 after applying unanimous voting. Similar observations were made across all applications of unanimous voting throughout the iterative labeling processes of three experiments, clearly demonstrating the effectiveness of unanimous voting in preventing the waste of review time.

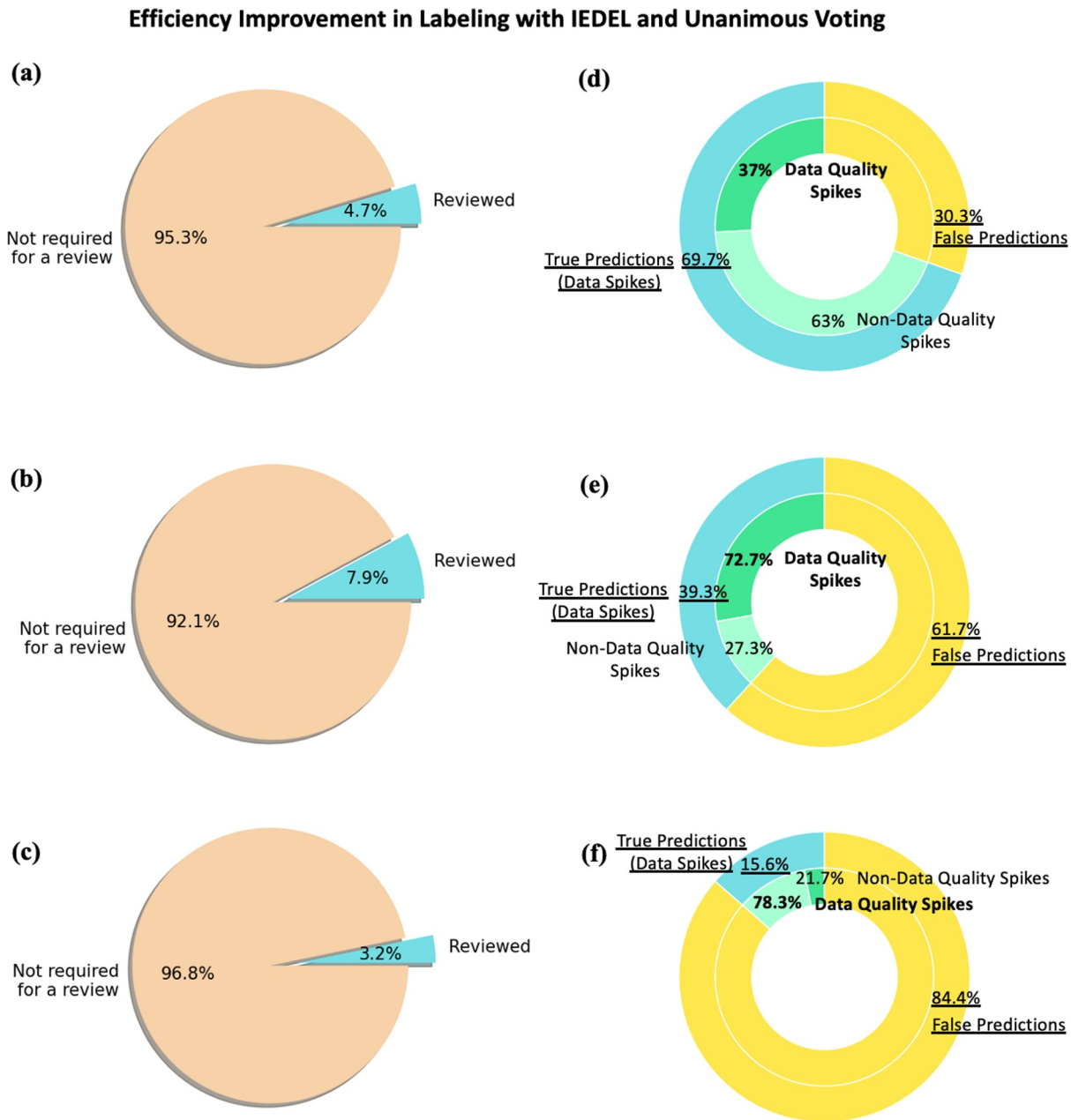

Figure 6. Evaluation of review efficiency improvement and precision using the Iterative Error-Driven Ensemble Labeling Algorithm. Panels (a), (b), and (c) show the data review percentages for the data sets "nsametC1.rh_mean," "sgpmetE37.rh_mean," and "sgpmetE37.temp_mean," respectively, illustrating the proportion of data manually reviewed versus that not required for review. Panels (d), (e), and (f) correspond to each respective data set and display the precision of the reviewed data, indicating the percentage of data spikes detected within the reviewed data.

4.3. Precision-Recall Trade-Offs

We observed precision and recall trade-offs in all three experiments when applying unanimous voting with IEDEL. Leveraging these trade-offs allows us to control the desired precision and recall at each iteration of IEDEL. In IEDEL, precision reflects the return on review efforts, while recall indicates how much we can increase the completeness of labeling true positive samples in a single iteration. For clarification, the term "recall" refers to an approximate estimate of the completeness of true positive labels, since we can't ensure a perfect accuracy of data labels without laboriously reviewing every single sample.

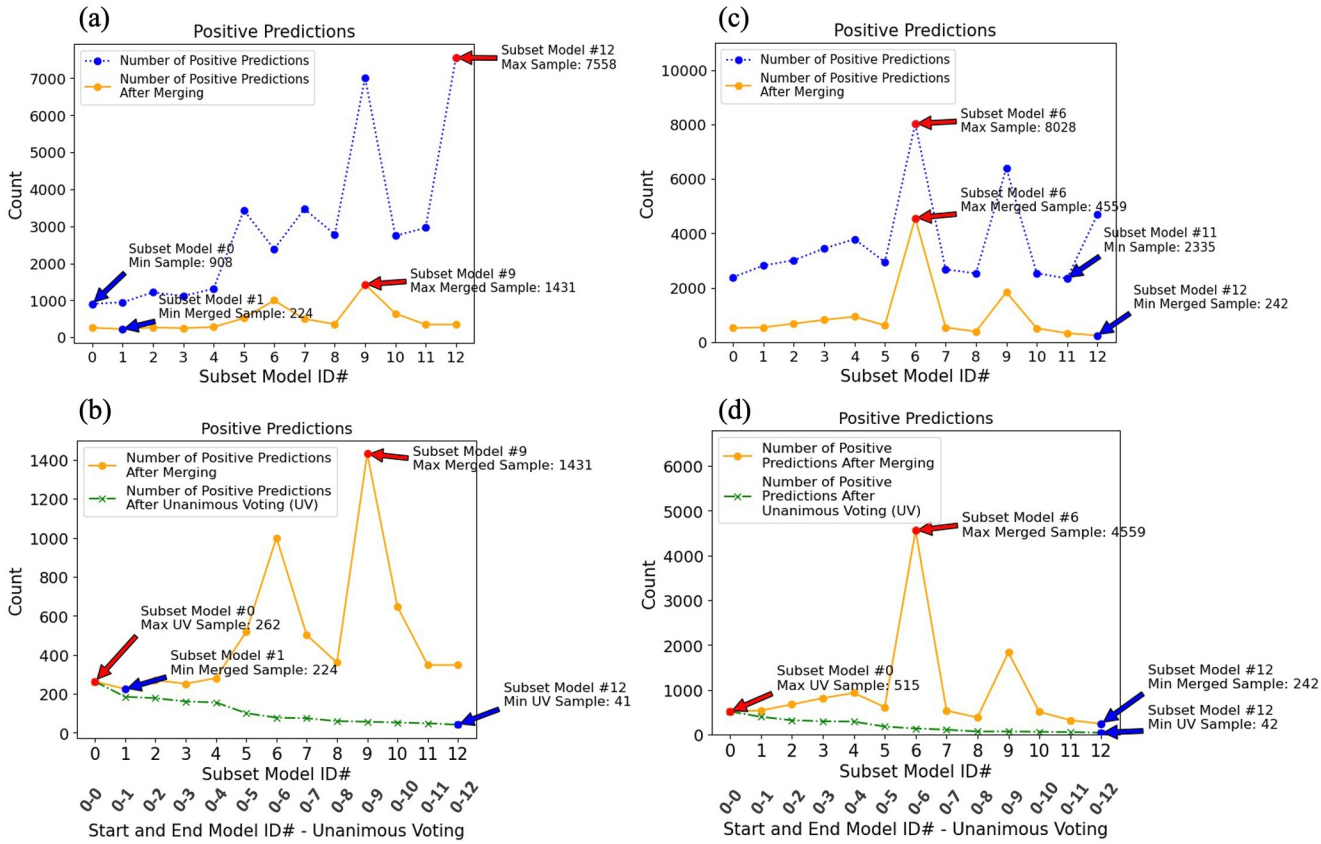

Figure 7. Reduction in review effort after applying Unanimous Voting in data set "nsametE37.rh_mean." Panel (a) and (b) present results from a model trained using data spanning 2019–2021, which was then applied to predict outcomes in data from 2013 to 2018. Panel (a) illustrates a comparison: it contrasts the initial count of positive predictions with the adjusted count obtained after merging overlapping samples. Panel (b) demonstrates the significant reduction in the number of positive predictions achieved through the implementation of unanimous voting algorithm. Panels (c) and (d) showcase results from a separate model, trained using data from 30 September 2020 to 11 December 2020. This model was utilized to generate predictions on data from 2019 to 2021, excluding the period used for training. The observations in Panels (c) and (d) mirror those in Panels (a) and (b).

We don't always want to apply unanimous voting to all subset models, as doing so results in the minimum number of positive predictions, leading to high precision but low recall. For instance, as shown in Figure 7, when we apply unanimous voting to all 13 subset models, we achieved 51% precision and 31% recall. However, by applying unanimous voting only to the first 8 models (from subset 0 to subset 7), we attain a precision of 44.6% and a recall of 48.5%, resulting in a more balanced outcome. This illustrates a trade-off, with a 6.4% decrease in precision for a 17.5% increase in recall.

In practice, while we won't be able to ascertain the exact trade-off between precision and recall at each iteration, we can still use this trade-off principle to determine what best suits our needs. For example, if we aim for a validation effort of 50 time intervals per iteration, we only need to continuously apply unanimous voting on subset models until the number of samples requiring review falls below 50. In this way, we can trade off a moderate decrease in precision for a higher recall, ultimately requiring fewer iterations to finish the whole IEDEL process.

4.4. Transfer Learning Performance

The implementation of transfer learning is crucial to the success of our project in leveraging ML algorithms to monitor data quality issues within the ARM Archive, since we started with insufficient labels. The IEDEL algorithm relies on a small set of labeled data to guide subsequent labeling iterations. However, manually labeling enough anomalies has proven to be a time-consuming and sometimes challenging task. To address this challenge, instead of laboriously searching for anomalies, we applied the transfer learning technique as detailed in Tables 1 and 2. This technique allows us to make predictions on unlabeled data and subsequently use these predictions to initiate the IEDEL implementation on new data sets.

Table 6
A Summary of Transfer Learning Performance Across Three Experiments in the Spike Detection Project

Transfer learning		Precision						Recall					
		IEDL				IEDEL & UV		IEDL				IEDEL & UV	
		Min	# Positive predictions	Max	# Positive predictions		# Positive predictions	Min	# Positive predictions	Max	# Positive predictions		# Positive predictions
1	Metrics	3.98% (Recall 89.71%)	1,431	22.32% (Recall 72.06%)	224	51.22%	41	67.65% (Precision 6.97%)	646	89.71% (Precision 3.98%)	1,431	30.88%	41
	Model ID	Model #9		Model #1		All		Model #10		Model #9		All	
	Precision & Recall per review	0.0028% 0.0627%		0.0996% 0.3217%		1.24% 0.7532%		0.0108% 0.1047%		0.0028% 0.0627%		1.24% 0.7532%	
2	Metrics	0.53% (Recall 89.66%)	4,559	8.68% (Recall 79.31%)	242	52.38%	42	72.41% (Precision 6.17%)	324	100.00% (Precision 4.81%, 3.85%, 3.19%, 2.79%)	541, 675, 816, 932	34.38%	42
	Model ID	Model #6		Model #12		All		Model #11		Model #1, 2, 3, 4		All	
	Precision & Recall per review	0.0001% 0.0197%		0.0359% 0.3277%		1.2471% 0.8186%		0.0190% 0.2235%		0.0049% 0.1350%		1.2471% 0.8186%	
3	Metrics	1.79% (Recall 88.89%)	4,692	14.04% (Recall 42.42%)	292	55.26%	26	30.30% (Precision 13.68%)	212	94.95% (Precision 7.48%)	1,203	14.14%	26
	Model ID	Model #6		Model #11		All		Model #12		Model #4		All	
	Precision & Recall per review	0.0004% 0.0189%		0.0481% 0.1453%		2.1254% 0.5438%		0.0645% 0.1429%		0.0062% 0.0789%		2.1254% 0.5438%	
4	Metrics	0.56% (Recall 69.70%)	3,780	7.51% (Recall 57.58%)	253	14.55%	55	42.42% (Precision 1.25%)	1,123	69.70% (Precision 0.56%)	3,780	24.24%	55
	Model ID	Model #F		Model #A		All		Model #G		Model #F		All	
	Precision & Recall per review	0.0001% 0.0184%		0.0297% 0.2276%		0.2645% 0.4407%		0.0011% 0.0378%		0.0001% 0.0184%		0.2645% 0.4407%	
Weighted average precision & Recall		1.29% 83.68%		12.96% 61.61%		39.86% 26.90%		4.69% 52.59%		2.73% 81.08%		39.86% 26.90%	
Weighted average precision & Recall per review		0.0001% 0.0058%		0.0128% 0.0609%		0.2430% 0.1640%		0.0020% 0.0228%		0.0004% 0.0113%		0.2430% 0.1640%	

We conducted an analysis to evaluate the performance of transfer learning data under various scenarios, as shown in Tables 6 and 7, including transfer learning over time, across extended facilities within the same ARM site, between different ARM sites, and across diverse variables. In the #1 transfer learning process, the maximum precision and a corresponding recall were 22.32% and 72.06% when reviewing 224 samples produced from subset model #1, and the maximum recall and a corresponding precision were 89.71% and 3.98% when reviewing 1,431 samples from subset model #9. These statistics highlight a performance trade-off in the IEDL algorithm between a decrease in precision by 18.34% and an increase in recall by 17.65%, and vice versa. The transfer learning performance of IEDL, while not impressive by itself, is better than unguided browsing for anomalies, especially

JGR: Machine Learning and Computation

10.1029/2024JH000192

Table 7
Sources and Time Ranges of Data Used in Each Transfer Learning Process for the Spike Detection Project

Transfer learning processes			From	To
1	**Temporal Transfer** within the Same Variable	Data set	nsametC1.rh_mean	nsametC1.rh_mean
		Time Range	2020/09/30–2020/12/11 (72 days)	2019/01/01–2021/12/31 (3 years, excluding 72 days used in training)
2	**Temporal Transfer** within the Same Variable	Data set	nsametC1.rh_mean	nsametC1.rh_mean
		Time Range	2019/01/01–2021/12/31 (3 years)	2013/01/01–2018/12/31 (6 years)
3	**Across Sites Transfer** within the Same Variable	Data set	nsametC1.rh_mean	sgpmetE37.rh_mean
		Time Range	2020/09/30–2020/12/11 (72 days)	2011/09/28–2023/09/29 (13 years)
4	**Across Variables Transfer** within the same facility	Data set	sgpmetE37.rh_mean	sgpmetE37.temp_mean
		Time Range	2011/09/28–2023/09/29 (13 years)	2011/09/28–2023/09/29 (13 years)

Note. This table offers additional details complementary to those in Table 6.

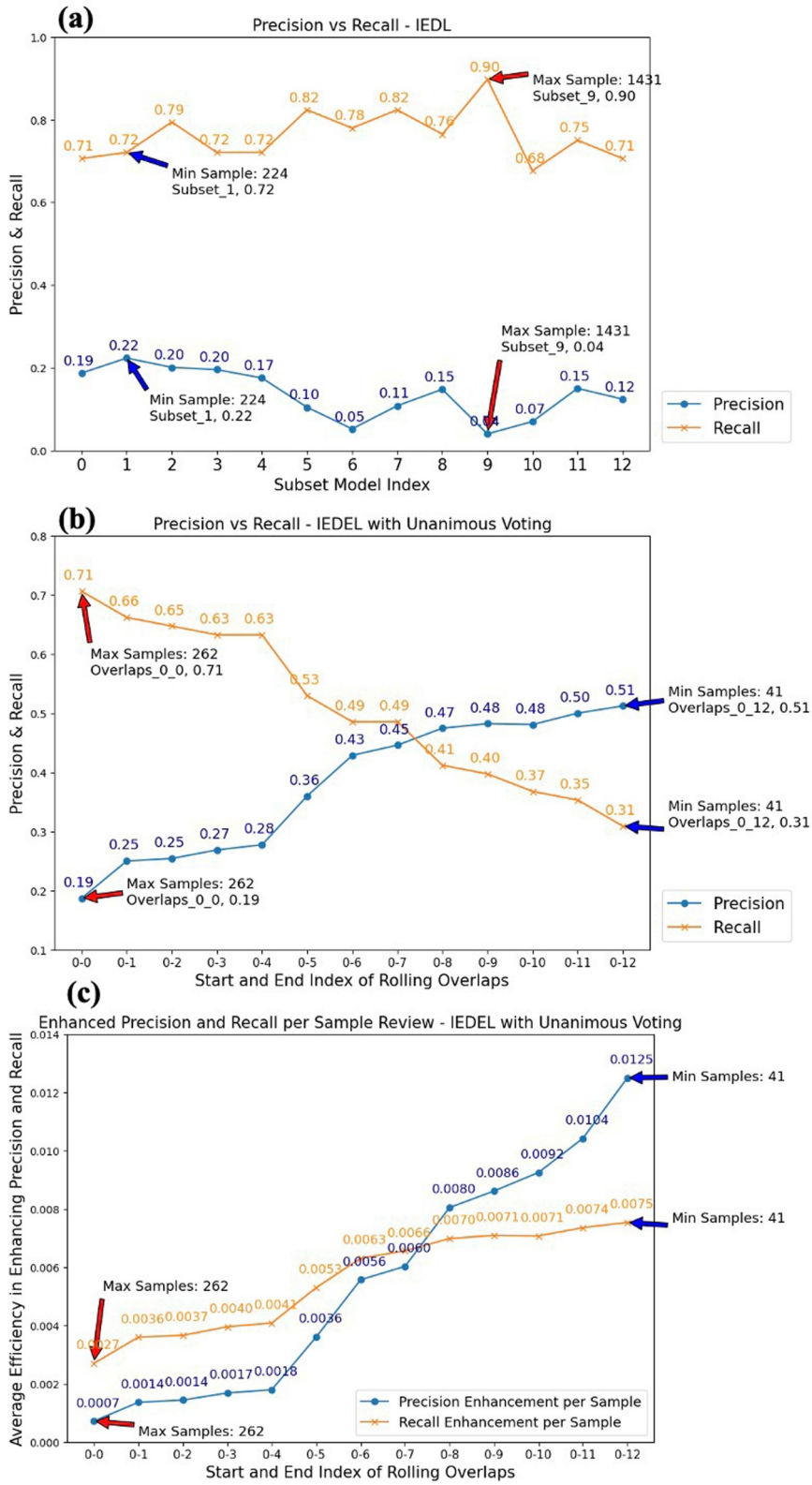

Figure 8. Comparative analysis of precision and recall for Iterative Error-Driven Labeling (IEDL) and Iterative Error-Driven Ensemble Labeling (IEDEL) algorithms: (a) performance metrics for IEDL, detailing precision and recall across different subset model indices with annotations of minimum and maximum sample sizes. (b) Delineates the same metrics for IEDEL, with annotations indicating the extent to which sample sizes were reduced through the application of IEDEL with Unanimous Voting. (c) Presents the efficiency in enhancing precision and recall per sample review when applying IEDEL with Unanimous Voting.

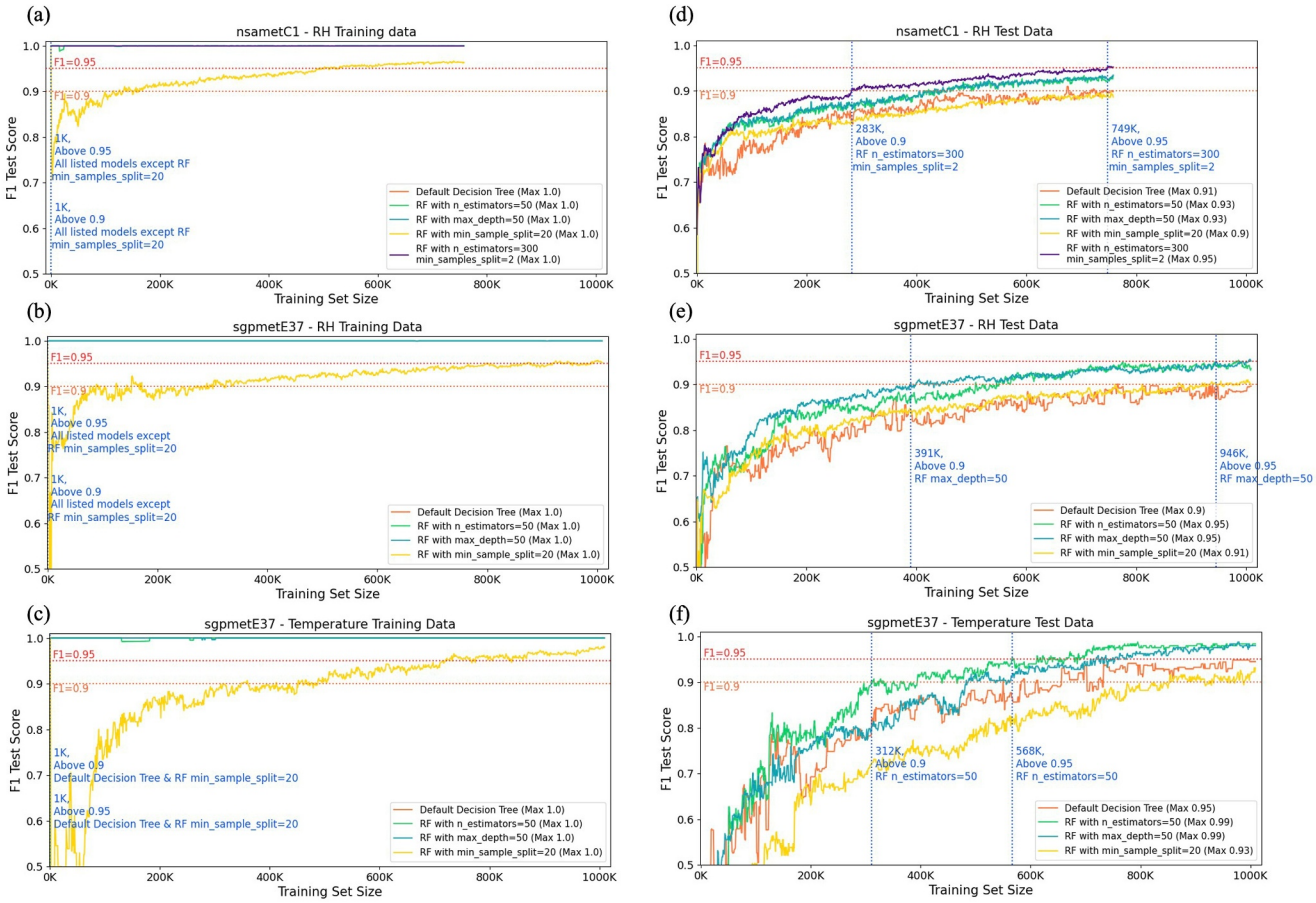

Figure 9. F1 score learning curves for model evaluation and selection. These panels correspond to F1 scores for six data sets: training data for (a) nsametC1.rh_mean, (b) sgpmetE37.rh_mean, and (c) sgpmetE37.temp_mean; and test data for (d) nsametC1.rh_mean, (e) sgpmetE37.rh_mean, and (f) sgpmetE27.temp_mean. Panels (a), (b), and (c) illustrate the trends in F1 scores on training data as the training data size increases, while panels (d), (e), and (f) showcase the trends estimated on the test data. Horizontal lines at F1 scores of 0.9 and 0.95 are marked, along with vertical lines indicating the points at which models first achieve these thresholds. The legend details the specific model settings employed for each data set.

considering that these results were achieved through transfer learning from 72 days of labeled data to 3 years of data. With iterative reviews of prediction errors, IEDL can gradually improve accuracy in labeling anomalies.

As shown in Table 6, the IEDEL algorithm with unanimous voting, compared to IEDL, improved transfer learning precision in the #1 process from a range of 3.98% to 22.32% to an aggregated result of 51.22%, in the #2 process from a range of 0.53% to 8.68% to an aggregated result of 52.38%, in the #3 process from a range of 1.79% to 14.04% to an aggregated result of 55.26%, and in the #4 process from a range of 0.56% to 7.51% to an aggregated result of 14.55%. Overall, the average weighted precision across all aforementioned transfer learning processes increased from a range of 1.29% to 12.96% to an aggregated result of 39.86%, trading off with a decrease in transfer learning recall from a range of 52.59% to 80.54% to an aggregated result of 26.9%. The effectiveness of this trade-off is possibly debatable considering the varying number of samples required for review across different aggregated results.

To further assess efficiency improvements in labeling from IEDL to IEDEL with unanimous voting, we estimated the average precision and recall contributed by each sample review. Although Figure 8b shows a trade-off between precision and recall, Figure 8c indicates a consistent increase in both metrics per review with unanimous voting implementation. The maximum weighted average precision and recall per review for IDEL across four processes listed in Table 6 were 0.0228% and 0.0609%, respectively. In comparison, IEDEL achieved 0.2430% and 0.1640% respectively, marking an efficiency enhancement by 2–10 times. These findings validate our proposal of the IEDEL algorithm and demonstrate that the IEDEL algorithm, with a bunch of models, outperforms the IEDL algorithm with a single model in enhancing the efficiency of labeling data anomalies.

While it remains uncertain which subset model can consistently produce predictions with the highest precision or recall, it is evident that utilizing IEDEL with unanimous voting can significantly enhance precision at each iteration. Consequently, the overall validation effort required for anomaly detection is substantially reduced.

4.5. Model Evaluation and Selection

Generally, by fine-tuning model hyperparameters through K-fold cross-validation, we can determine the optimal model for our specific use case. However, when multiple models demonstrate strong performance on both the training and test data sets, hyperparameter tuning alone may no longer provide valuable insights for model selection. To select the best model for production, our primary criteria is the model's ability to generalize accurately to future data, rather than just its training performance. Therefore, we propose using a learning curve to evaluate how a model's performance might evolve with more data, simulating an assessment of the model's current performance versus its future performance.

As illustrated in Figure 9d, the Random Forest model with "n_estimators" set to 300 and "min_samples_split" set to 2 archives the highest F1 score on test data for the "nsametC1.rh_mean" data set, reaching an F1 score above 0.9 when 283,000 training samples are used in model training, and above 0.95 when 749,000 samples are used, indicating that this model is clearly the best performer among the five models evaluated.

However, model selection can sometimes be challenging. As shown in Figure 9e, a default decision tree and a random forest with "min_samples_split" set to 20 are closely matched, with only a 0.0043 difference in F1 scores when the entire training data set is utilized for model training. Selecting a model between these two based solely on their performance on test data would be contentious, as minor variations in data splitting or randomness of model configuration could reverse this decision. But, when examining the learning curves of both models, it becomes evident that the random forest with "min_samples_split" set to 20 demonstrates more consistent performance improvement and overall outperforms the default decision tree.

Additionally, the differences in F1 scores between training and test data for the best model performance across three scenarios range from 1% to 5%, indicating that these final models are not overfitting the training data. More details on precisions and recalls can be found in the Figure S5 in Supporting Information S1.

5. Conclusions

In this paper, we introduce the "IEDEL with Unanimous Voting" algorithm, an innovative labeling method tailored for anomaly detection in ARM's data. This method integrates transfer learning and iterative reviews of prediction errors, enabling the development of robust ML models that we can deploy to complement existing QC tests within ARM data sets. Besides, by incorporating transfer learning, IEDEL can be effectively scaled across different datastreams in the ARM Archive. IEDEL is versatile, suitable for addressing various types of data anomalies using a consistent process. The primary distinctions across different anomaly types primarily lie in the process of data preprocessing and feature engineering. It's crucial to note that while IEDEL does not guarantee perfect label accuracy, it can achieve sufficient accuracy to support the development of effective ML models for data monitoring purposes.

Our upcoming DQO plan, which focuses on leveraging ML algorithms for data QC, will be dedicated to establishing a framework to operate ML pipelines. This includes implementing IEDEL with Unanimous Voting and devising strategies for data preparation and features engineering specific to each type of data anomaly. For example, in our spike detection project, we use a time window of 20 steps, with a stride of 1 for minority class samples and 5 for majority class samples. In contrast, drift detection should involve examining long-term data drifts, necessitating larger time windows and strides. In summary, IEDEL presents a promising avenue for employing supervised learning algorithms in the data quality monitoring of ARM Program data.

Data Availability Statement

The Atmospheric Radiation Measurement (ARM) user facility data used for spike detection experiments in the study are available at the ARM Data Discovery Website via https://doi.org/10.5439/1786358.

Acknowledgments
Funding was provided by NOAA/Office of Oceanic and Atmospheric Research under NOAA—University of Oklahoma Cooperative Agreement #NA21OAR4320204, U.S. Department of Commerce. This research was supported by the Atmospheric Radiation Measurement (ARM) user facility, a U.S. Department of Energy (DOE) Office of Science user facility managed by the Biological and Environmental Research Program.

References

ARM Standards Committee. (2020). ARM data file standards version: 1.3. pdf. Retrieved from https://www.arm.gov/publications/programdocs/doe-sc-arm-15-004.pdf

Божиновски, А. (2020). Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica*, *44*(3), 291–302. https://doi.org/10.31449/inf.v44i3.2828

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. *SIGMOD Record*, *29*(2), 93–104. https://doi.org/10.1145/335191.335388

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3), 1–58. Article 15. https://doi.org/10.1145/1541880.1541882

Chawla, N., Bowyer, K., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. ArXiv, abs/1106.1813.

Giannoni, F., Mancini, M., & Marinelli, F. (2018). Anomaly detection models for IoT time series data. *arXiv e-prints*. arXiv:1812.00890.

Iliopoulos, A., Violos, J., Diou, C., & Varlamis, I. (2023). Detection of anomalies in multivariate time series using ensemble techniques.

Kyrouac, J., Shi, Y., & Tuftedal, M. (2011). Surface Meteorological Instrumentation (MET). Atmospheric Radiation Measurement (ARM) user facility. https://doi.org/10.5439/1786358

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining* (pp. 413–422).

Markou, M., & Singh, S. (2003). Novelty detection: A review—Part 1: Statistical approaches. *Signal Processing*, *83*(12), 2481–2497. https://doi.org/10.1016/j.sigpro.2003.07.018

Nayeem Rizve, M., Duarte, K., Rawat, Y. S., & Shah, M. (2021). Defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. arXiv:2101.06329.

Oliver, A., Odena, A., Raffel, C., Cubuk, E. D., & Goodfellow, I. J. (2018). Realistic evaluation of deep semi-supervised learning algorithms. In *Neural information processing systems*.

Peppler, R. A., Kehoe, K. E., Monroe, J. W., Theisen, A. K., & Moore, S. T. (2016). The ARM data quality program. *Meteorological Monographs*, *57*(1), 12.1–12.14. https://doi.org/10.1175/AMSMONOGRAPHS-D-15-0039.1

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*(7), 1443–1471. https://doi.org/10.1162/089976601750264965

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017). DBSCAN Revisited, Revisited: Why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems*, *42*(3), 1–21. Article 19. https://doi.org/10.1145/3068335

Sisterson, D. L., Peppler, R. A., Cress, T. S., Lamb, P. J., & Turner, D. D. (2016). The ARM Southern Great Plains (SGP) site. *Meteorological Monographs*, *57*(1), 6.1–6.14. https://doi.org/10.1175/AMSMONOGRAPHS-D-16-0004.1

Turner, D. D. & Ellingson, R. G. (Eds.). (2016). *The Atmospheric Radiation Measurement (ARM) program: The first 20 years.* Meteorological Monographs, 57(1). Retrieved from https://journals.ametsoc.org/view/journals/amsm/57/1/amsm.57.issue-1.xml

U.S. Department of Energy. (1998). Atmospheric Radiation Measurement (ARM) user facility data are collected through routine operations and scientific field experiments. Retrieved from https://www.arm.gov/data/

Verlinde, J., Zak, B. D., Shupe, M. D., Ivey, M. D., & Stamnes, K. (2016). The ARM north slope of Alaska (NSA) sites. *Meteorological Monographs*, *57*(1), 8.1–8.13. https://doi.org/10.1175/AMSMONOGRAPHS-D-15-0023.1

Xu, Q., Likhomanenko, T., Kahn, J., Hannun, A. Y., Synnaeve, G., & Collobert, R. (2020). Iterative pseudo-labeling for speech recognition. In *Interspeech*.

Yu, B., Pan, D. Z., Matsunawa, T., & Zeng, X. (2015). Machine learning and pattern matching in physical design. In *The 20th Asia and South Pacific Design Automation Conference* (pp. 286–293).

Erratum

The originally published version of this article omitted a funding source. The Acknowledgments have been revised to read as follows: "Funding was provided by NOAA/Office of Oceanic and Atmospheric Research under NOAA—University of Oklahoma Cooperative Agreement #NA21OAR4320204, U.S. Department of Commerce. This research was supported by the Atmospheric Radiation Measurement (ARM) user facility, a U.S. Department of Energy (DOE) Office of Science user facility managed by the Biological and Environmental Research Program." The error has been corrected, and this may be considered the authoritative version of record.