Active learning for imbalanced data under cold start

Ricardo Barata
ricardo.barata@feedzai.com
Feedzai

Miguel Leite*
miguel.leite@feedzai.com
Feedzai

Ricardo Pacheco*
rjgpacheco@gmail.com
Feedzai

Marco O. P. Sampaio
marco.sampaio@feedzai.com
Feedzai

João Tiago Ascensão
joao.ascensao@feedzai.com
Feedzai

Pedro Bizarro
pedro.bizarro@feedzai.com
Feedzai

ABSTRACT

Modern systems that rely on Machine Learning (ML) for predictive modelling, may suffer from the *cold-start* problem: supervised models work well but, initially, there are no labels, which are costly or slow to obtain. This problem is even worse in imbalanced data scenarios, where labels of the positive class take longer to accumulate. We propose an Active Learning (AL) system for datasets with orders of magnitude of class imbalance, in a cold start streaming scenario. We present a computationally efficient Outlier-based Discriminative AL approach (ODAL) and design a novel 3-stage sequence of AL labeling policies where ODAL is used as warm-up. Then, we perform empirical studies in four real world datasets, with various magnitudes of class imbalance. The results show that our method can more quickly reach a high performance model than standard AL policies without ODAL warm-up. Its observed gains over random sampling can reach 80% and be competitive with policies with an unlimited annotation budget or additional historical data (using just 2% to 10% of the labels).

CCS CONCEPTS

• **Computing methodologies** → **Active learning settings**; *Supervised learning by classification*; **Online learning settings**;

KEYWORDS

active learning, data streams, cold start, high class imbalance

ACM Reference Format:
Ricardo Barata, Miguel Leite, Ricardo Pacheco, Marco O. P. Sampaio, João Tiago Ascensão, and Pedro Bizarro. 2021. Active learning for imbalanced data under cold start. In *Proceedings of ACM International Conference on AI in Finance, Nov 2021.* ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

Training high performance supervised Machine Learning (ML) models is currently an essential and widespread task in the digital domain, where vast amounts of data are generated daily in numerous applications (e.g., finance, entertainment or consumer goods services). Those models are often central in decisions that enhance system efficiency, user experience or even safety and they rely heavily on collecting high quality labeled data. In many use cases, labeled data is initially absent (*cold start*) and expensive to collect, often requiring human annotation under a limited budget, while it is common that the system collects large amounts of unlabeled data. Thus, as data arrives and accumulates in the system, it becomes essential to select the most informative samples for labeling to quickly be able to train a high performance ML model – this is the goal of Active Learning (AL), as represented in Figure 1.

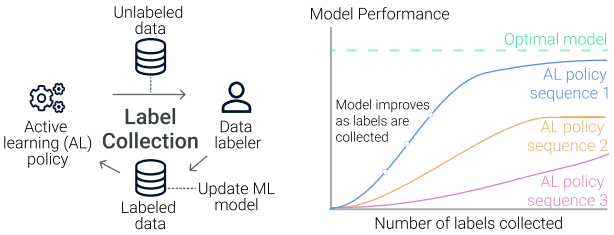

Figure 1: Schematics of the AL loop (left) and model performance evolution (right) depending on the AL policy.

In this paper, we propose an AL-based annotation method for real-time data streams with a large class imbalance, to train a high performance model in a *cold start* scenario. We perform an extensive empirical study using real world datasets of credit card transactions where the ML task is to detect fraudulent transactions. AL is especially relevant in this domain, since there is, usually, a considerable delay between the fraudulent event and the collection of the label (e.g., through client complaints or reports from financial institutions) unless a human analyst is consulted. We test well known AL policies, as well as our proposed sequences of policies that are especially designed for imbalanced datasets, to achieve a high performance, with reduced variance, in few iterations. Our main contributions are:

- A new computationally efficient approach to the Discriminative Active Learning method [6] named Outlier based Discriminative Active Learning (ODAL) – Section 2.2.2.
- Two variations of uncertainty sampling policies using an epistemic uncertainty measure, as well as a measure based on the fraud rate percentile – Section 2.2.3.
- A 3-stage sequence of policies suited to highly imbalanced datasets, using ODAL in the second stage (*warm-up* policy) – Section 2.2.4.
- An extensive set of experiments to identify the best AL setup for fraud detection on four real world datasets – Section 4.

*Work developed while employed at Feedzai.

ICAIF '21, November 3–5, 2021, Online

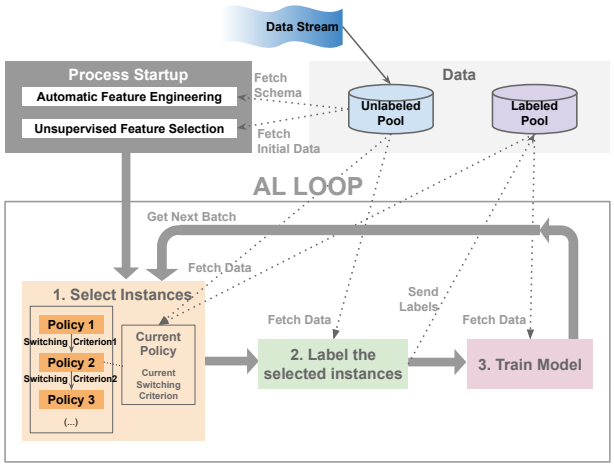

Figure 2: Experimental framework architecture overview.

2 METHODS

In this section we present the *use case* we will study in our experiments together with a description of the methods tested. Our basic problem consists of prioritizing instances from a growing unlabeled pool of data in a streaming environment for labeling (pool based sampling [18]) using AL. The number of instances in one labeling query request (henceforth *query*), i.e., the *batch size*, is a parameter that may influence how fast AL improves the ML model trained with the collected labels. We focus on a realistic streaming data *cold start* scenario, where initially, there is no labeled or unlabeled data available but rapidly unlabeled data accumulates. In particular, when AL selects a batch of queries for labeling, at a given iteration, more unlabeled data will be available than on previous iterations.

With this streaming data setup in mind, we present an illustrative diagram of the architecture of our full framework in Figure 2 that will serve as reference when explaining the methods we developed. Its main components are:

- **Data Components**: This contains a *Data Stream* collecting events in real time and storing them in the *Unlabeled pool*. The *Labeled pool* stores labeled data. Both pools start empty.
- **Process Startup**: This is responsible for training pre-processing pipelines, enriching the raw incoming data stream with features and applying feature selection and/or dimensionality reduction.
- **AL loop**: This iteratively collects labels and trains the model. At each step the *Data* is accessed and manipulated as follows:
  (1) *Select Instances*: A batch of unlabeled events is selected by the currently active policy for querying. An arbitrary sequence of AL policies chained together with switching criteria is possible (left of block 1), though in our experiments we only consider up to 3-stage sequences.
  (2) *Label Instances*: Here we simply move the instances selected for labeling from the *Unlabeled* to the *Labeled* pool and reveal their label (our data sources contain the true label). In a live system, analysts would provide the labels.
  (3) *Train Model*: The labeled data is used to train and evaluate the ML model. We continuously iterate this loop up to a maximum fixed duration (e.g., until a fixed number of labels is collected). Since we use historical data to simulate the data stream, we can evaluate the sequence of AL models, obtained while iterating (Figure 1), on a separate test set – see Section 3.

	Cold stage	Warmup stage	Hot stage
Baseline	QueryAll	–	–
1-stage	OutlierDetect	–	–
	Random	–	–
2-stage	Random	–	Unc. (entropy)
	Random	–	ODAL
	Random	–	EMC
	Random	–	QBC
3-stage	Random	ODAL	Unc. (entropy)
	Random	ODAL	Unc. (epistemic)
	Random	ODAL	Unc. (percentile)
	Random	ODAL	QBC
	Random	ODAL	EMC

Table 1: Policy sequences tested in experiments.

2.1 Startup and preprocessing

In a *cold start* scenario we may not know, in advance, which features are useful to predict the target. Thus, we apply a minimal preprocessing depending only on the schema of the data fields collected by the system. The transformations applied are now described.

Automatic Feature Engineering: We use Feedzai's AutoML tool [12], which generates a feature engineering plan based only on raw data fields. This only requires a file specifying, e.g., grouping entities, numerical fields, or the semantics of fields to be used in pre-defined types of feature engineering operations, together with a specification of window durations to compute profile feature aggregations (e.g., count of transactions per card in the last hour).

Unsupervised Feature Selection: From both a data science and system performance perspective it is useful to discard redundant or noise features produced by AutoML. We use *Principal Component Analysis* (PCA) [23] to reduce the dimensionality of the numerical features space by selecting the top principal components that explain a fraction of the variance in the data. This method requires a sample of unlabeled data, which can be easily collected through an initial waiting period (e.g., we use one day in our experiments).

2.2 Policies

The central ingredient in an AL based annotation system is the policy determining which instances are the most relevant to label. We categorize the types of policies to mirror our three-stage strategy to efficiently train a model from *cold start*:

(1) **Cold policies** (unsupervised): In a first stage, while no labeled data is available, a method is used to select the first instances for labeling before supervised AL can start – Section 2.2.1.
(2) **Warm-up policies**: After some labels are collected, there may be a transient period with only labels of a given type available (e.g., only negative class for binary classification) or too few labels to train a supervised policy – Section 2.2.2.
(3) **Hot policies** (supervised): These are the most common, and they make full use of the collected labels to differentiate classes and select the best instances to query – Section 2.2.3.

Next we discuss our choices for each stage. The combinations used in our experiments are summarized in Table 1.

2.2.1 Cold policies: AL studies in the literature often assume that a labeled pool is available to start the AL process. However, in many real world scenarios one may be faced with a system that has just been deployed and contains no labeled data [7]. Then, the initial sampling can only be guided by unlabeled instances. The simplest choice is to randomly sample an initial batch of instances – *Random Policy*. Another simple option is to use an unsupervised learning method to build a representation of the unlabeled data and select outliers – *Outlier Detection Policy* – which is useful if one or more of the classes behave as outliers. We test, as 1-stage baselines (Table 1), the *Random* policy as well as an *Outlier Detection* Policy consisting of an isolation forest [11] trained on the unlabeled pool. The isolation score is used to rank the unlabeled transactions from most outlier-like (to query) to most inlier-like. Experiments using this method will be identified with the tag *OutlierDetect. Cold* policies are also baselines for AL if used alone (i.e., one-stage sequence experiments).

2.2.2 Warm-up policies & ODAL. Regarding *warm-up* we propose a new method, *Outlier Discriminative Active Learning* (ODAL), that provides a computationally lighter approach to *Discriminative Active Learning (DAL)* [6]. The latter is based on the principle that a good labeled pool should be difficult to discriminate from the unlabeled pool. In this approach, a binary classification model is fit to discriminate between pools, then the unlabeled pool is scored and instances that are easy to discriminate from the labeled pool are queried. This can be computationally heavy because it always trains on all available data (labeled and unlabeled). In ODAL we propose, instead, to train an outlier detection algorithm on the labeled pool, and use the obtained model to score the unlabeled pool and find the greatest outliers relative to the labeled pool. Those instances are then queried for labeling. In typical AL scenarios the labeled pool is much smaller than the unlabeled pool, so ODAL can be trained on the labeled pool only, in contrast with DAL. Another advantage of ODAL over DAL is observed by expanding the DAL score, $p(0|x)$, for an instance with features x to be in the unlabeled pool (0) using Bayes theorem (assuming a probabilistic discriminator):

$$p(0|x) = \frac{p(x|0)p(0)}{p(x|0)p(0) + p(x|1)p(1)} = \left(1 + \frac{p(x|1)p(1)}{p(x|0)p(0)}\right)^{-1} . \quad (1)$$

Here $p(x|0)$, $p(x|1)$ are, respectively, the distributions of the unlabeled and labeled pools and $p(1) = 1-p(0)$ is the fraction of labeled data. In Eq. (1) we see that the DAL score is high for instances with a low density ratio, $p(x|1)/p(x|0)$, between the labeled and unlabeled pool, which may not be desirable if the labeled pool is missing examples in lower density regions of the unlabeled pool. On the other hand ODAL only models $p(x|1)$ so it favours, by design, that the instances to be selected are not well represented in the labeled pool regardless of how well they are represented in the unlabeled pool. For problems with a large class imbalance this is especially important. Thus, ODAL is both computationally feasible for our large scale experiments and less biased by the unlabeled data distribution. We will see in Section 4, that ODAL warm-up adds an early boost to the learning curves in imbalanced datasets. In the experiments, we will use an isolation forest outlier detection algorithm. Thus, the labeled pool instances will be ranked by isolation score and the ones ranking high are selected for labeling.

Within warm-up policies, there is another class of methods denoted *Density-weighted* that aim to select instances to cover well the most dense areas of the data distribution [5, 14, 24]. These methods tend to be heavier and harder to implement in streaming, because the unlabeled pool may grow and its distribution may drift in real-time, so we leave them out of our experiments.

2.2.3 Hot policies. We now describe the supervised policies.

Uncertainty Sampling: This is the most common active learning technique, originally discussed by Lewis and Gale [10]. It trains a machine learning model on each AL iteration using the labeled pool instances. Then the unlabeled pool is scored and the instances are ranked by a measure of uncertainty related to the distance to the classification boundary. Instances closer to the classification boundary are assumed to be more likely to improve the model. A common criterion is to select instances with the highest expected entropy over the possible class labels given the model scores as the probabilities. For binary classification those instances have scores closest to 0.5. This method assumes scores that are well calibrated probabilities, which may not hold. Despite studies showing that it is an efficient AL uncertainty measure ([25] and references therein), the calibration assumption may not work for many algorithms and it can be especially bad for high class imbalance [16].

We introduce an alternative for binary classification, that does not require score calibration. We first observe that the score function of most ML algorithms is a monotonic function of the class posterior probability. Thus we still expect that instances with higher scores will have a higher probability of being positive. Given a sample of data, the classification boundary can be equivalently characterized by a score percentile, i.e., a position in the sorted set of scores. We then note that the percentile of the classification boundary, for a perfect classifier that knows the labels would be equal to the negative class rate. This motivates an alternative uncertainty criterion, where the uncertain instances are the ones closest to the estimated negative class rate boundary. In the experiments, we will show results with the classic *entropy* criterion (denoted *Unc. (entropy)*), as well as with our *fraud percentile* criterion (denoted *Unc. (percentile)*).

Expected variance Reduction and Epistemic Uncertainty: The expected variance reduction method estimates the variance of the model predictions [4]. This is tightly related to the notion of epistemic uncertainty discussed in the literature [21]. Epistemic uncertainty is the reducible part of the total uncertainty composed of i) the model uncertainty (or bias), plus ii) the approximation uncertainty (variance). The remaining uncertainty (also know as aleatoric) is intrinsic to the data generating process and can never be removed. The standard uncertainty sampling criterion using the entropy of the model scores is precisely the total uncertainty criterion. The epistemic uncertainty, being the difference between the total and aleatoric uncertainty, may give a better measure of uncertainty for AL, because it is only sensitive to the reducible components. Though it still contains the uncertainty from the bias, it can be more tractable than variance estimates, which often rely on analytic expressions assuming differentiability. In our analysis, we train models using a random forest classifier. This is non-differentiable

but it offers a convenient way of controlling regularization, using a large number of shallow trees, which is important to train on small labeled pools. The epistemic uncertainty for random forests is estimated from the outputs of each tree, [21]. In our experimens, we denote the epistemic uncertainty criterion by *Unc. (epistemic)*.

Query By Committee (QBC): Query by committee [20], is a simple but potentially computationally heavier method that combines knowledge from an ensemble of ML models, chosen by the user, where each model in the ensemble is trained on the labeled data pool and used to score the unlabeled data. A measure of disagreement among the models is computed for each instance based on the model scores. Instances rank higher for higher disagreement. Often, it also assumes that the scores are well calibrated probabilities. Thus, here we introduce an alternative measure of disagreement, among the models in the committee, that is insensitive to whether or not the scores output by each model are calibrated as probabilities. This can be important if the committee contains a mixture of models with and without a probabilistic outcome. For each model in the committee, we rank the unlabeled instances by descending model score and compute the average pairwise absolute difference of ranks between any two models. Instances on which the models disagree will have very different rankings across models.

Expected Model Change (EMC): The basic principle of this method [19] is to query the instance that is expected to change the model the most. We use the simplest approach in the literature where: i) a gradient-based classifier is trained on the labeled data pool, ii) for each unlabeled instance, the expected gradient norm for the given instance is computed assuming that the model parameters are near an optimum of the model's loss function and, iii) the unlabeled pool instances are ranked so that instances with larger expected gradient are prioritized. A related method that is often impractical is *Expected Error Reduction* [17], which requires retraining the model for all label assignments for each possible query.

2.2.4 Policy Sequences. In addition to the AL policy sequences displayed in Table 1 we also add a baseline denoted *QueryAll* corresponding to unbounded labeling resources where all incoming transactions are labeled. In the 2-stage sequences we switch policies after we have at least one label from each class. Regarding the 3-stage sequences, the same criterion is used to switch between *warm-up* and *hot* policy, however, the switch to the *warm-up* policy from the *cold* policy is done after the first batch is collected with the *cold* policy, to start exploiting ODAL immediately.

3 EXPERIMENTS

In this section we present results of experiments with several real world credit card fraud datasets.

3.1 Policy Parameters

We make the following choices for the various policies in the experiments. In all policies that require an isolation forest we use the SCIKIT-LEARN [15] implementation with 100 trees, using all features to grow each tree, and a maximum number of samples per tree which is the minimum between 256 and the total number of samples. For *QBC* we use a committee with: a Random Forest with 100 trees and maximum depths of 3, an L2 regularized Logistic Regression, a Gaussian naive Bayes classifier, and a Gradient Boosting Classifier with 100 estimators.[1] For *EMC* we use a logistic regression with L2 regularization.

As for the batch size, we use 100 for all policies. In preliminary experiments with smaller batch sizes we did not see substantial improvements, while larger sizes degrade the results.

Dataset	Positive class rate	Sampling fraction
Bank 1	$\sim 10^{-4}$	11.0 %
Bank 2	$\sim 10^{-3}$	3.0 %
Payment Processor	$\sim 10^{-2}$	2.5 %
Merchant	$\sim 10^{-2}$	100.0 %

Table 2: *Dataset properties:* Due to privacy reasons we do not provide further details (see detailed description in text).

3.2 Data preparation

We cover several representative use cases in the fraud detection domain, namely card issuing banks (Banking), platforms that process online payments for several merchants (Payment Processors) and single merchant online platforms (Merchants).

In Table 2 we provide some properties of each data set, which contain fraudulent (positive) and legitimate (negative) transactions. The fraud rates span several orders of magnitude, from an extremely large imbalance (Bank 1), to moderate imbalances of a few percent.

The datasets contain raw fields collected when transactions arrived to a fraud detection system in real-time, including the monetary amount of the transaction, the timestamp of the event, several identifiers (e.g., card ID), categorical fields and the fraud label.

The volume of transactions varies across datasets from a few millions to several hundreds of million per year. To speed up our experiments, we applied undersampling to reduce the volume to a manageable (and similar) level for all datasets. This allowed us to scale up our experiments to cover many different types of policies and to perform a more extensive Temporal Cross Validation (TCV) over a longer period. We applied the sampling before feature engineering to speed up the preprocessing. Fraudulent and non-fraudulent card id entities were randomly sampled separately with the sampling rate indicated in Table 2 (this preserves the fraud rate) and all transactions were kept for each sampled card id. This keeps complete card histories, allowing to compute sliding window profiles that are important to characterize the event [1].

We applied automatic feature engineering, which generated between 600 and 800 features depending on the dataset – see Section 2.1. The categorical fields were encoded both with ordinal and frequency encoding and standardized to zero mean and unit variance similarly to other numerical features. The remaining preprocessing is scenario specific – details provided in Section 3.3.

3.3 Experimental Setup

In this section we describe details of the experimental setup that are common to all data sets.

3.3.1 Data slicing. In Figure 3 we present a diagram of the various slices of data for any given data set. We define *Folds*, which consist of 8 week periods (two pairs of 4 weeks). Within each fold, the first 4 weeks (green), are used for model training, whereas the following

[1] We use SCIKIT-LEARN [15] implementations for all mentioned ML models unless stated otherwise. For the unspecified hyper-parameters we use the library defaults.

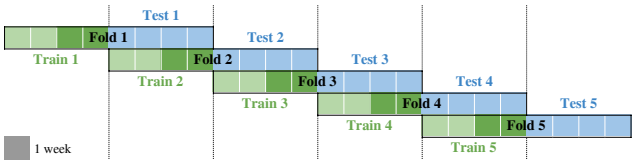

Figure 3: Time folds for the five simulation periods in the experiments (see detailed description in the text).

4 weeks (blue), are for model evaluations. The *Train* period is used differently according to the type of experimental run.

3.3.2 AL scenario and baselines. Here we describe the details of data preprocessing pipeline preparation and training periods.

AL in streaming. This case mimics a scenario where the AL system is deployed for the first time in streaming without access to previous data. Since the goal is to collect labels quickly to obtain a good model, without waiting for labels to arrive by other means, applying AL is typically relevant for a few weeks. Thus, we only reserve the two last weeks of the *Train* periods (darker green: weeks 3 and 4) to sample data with AL for training (weeks 1 and 2 are used for the strong optimistic baseline discussed next). The *Test* sets allows us to measure the model performance after the deployment of the last AL model. In practice, for most data sets we only use one week for AL training (except for Bank 1 which, due to the extreme class imbalance, needs a longer period for the performance to stabilize).

Optimistic Baseline. Here we train a strong model that has access to all *Train* data and labels (weeks 1 to 4: light plus dark green). The goal is to obtain a "best case scenario" upper bound performance.

3.3.3 Training procedure. Each experiment (either AL or Optimistic Baseline) consists of 35 repetitions of the train-test procedure with different pseudo-random number generator seeds. This allows us to assess the stability of the AL policies by observing the variance of our metrics. We choose 35 seeds as a good trade off between run time and a high chance of observing a wider range of values around the center of the distribution. As displayed in Figure 3, we repeat each experiment in 5 different folds (*Train*+*Test* pairs) to observe the robustness of the AL procedure against temporal variations.

Streaming AL Training. In all AL experiments we include an initial waiting period of one day to simulate the collection of some unlabeled data to fit the pre-processing pipeline. This mimics a realistic scenario of deployment with no previous data. To reduce the number of numerical features generated by the AutoML pipeline (which may contain redundant information) we apply PCA on the numerical features. In preliminary experiments on Bank 2, we checked that about 90 features can explain 99% of the data variance. Then we decided to fix 90 features after PCA for all data sets to keep the run time similar across experiments.

Observe that our pre-processing pipeline is trained on the first day of unlabeled data, and used to transform all future data arriving at the stream (*Train* or *Test* period). This is to mimic a day-1 system deployment. However, after day-1, the pipeline could be updated frequently but, for simplicity, we chose to fix it in our experiments.

For each run, several labeling iterations are processed after the waiting period of one day, according to the diagram of Figure 2 – see Section 2. Therefore the unlabeled pool grows with time, as does the labeled pool during the AL training iterations, whose growth is indirectly controlled by the time assumed for the team of analysts to label each queried batch of events. Thus, if the team is, e.g., a single analyst taking one hour to review a batch, we assume that one hour of new data is inserted in the unlabeled pool after the batch is labeled. For simplicity we use a fixed batch size and a fixed time to review corresponding to an overall review rate of 1000 events per day. The only exception is for Bank 1, where, due to the extreme class imbalance, we assumed twice the daily budget.

Regarding the ML model to train on the AL labeled data, we chose a highly regularized Random Forest (RF) classifier from the SCIKIT-LEARN library with a maximum tree depth of 3 and 200 trees (other hyper-parameters set to defaults). We did a small study on Bank 2 on two time folds, where we either, i) varied the number of trees up to 1000, ii) reduced or increased the maximum depth, or iii) used other models with various different levels of regularization (Feed Forward Neural Networks, Support Vector Machines and Naïve Bayes). This confirmed the benefits of regularization. Despite improvements with 1000 trees, we chose 200 to speedup our simulations.

Optimistic Baseline Training. Here we assume access to fully labeled data in the 4 weeks of the *Train* period. Additionally we apply a more robust training methodology. We train a RF classifier with 300 trees and a maximum depth of 20. For each of the 35 models (one per seed) we train 5 random configurations of hyper-parameters on the first 3 weeks and evaluate on week 4 to select the best configuration. The final configuration is re-fit on the 4 weeks.

For each model trained above, we also apply supervised feature selection. The fraction of features to use is a hyper-parameter to vary. In addition, we also vary the minimum number of samples in a leaf node, a binary parameter (to use class weights or not), and the complexity parameter for minimal cost-complexity pruning.

3.3.4 Evaluation metrics. We now discuss the performance metrics used to measure the quality of a single AL experiment, as well as to aggregate and summarize an experiment to compare runs.

Learning curves. A single AL experiment, consists of several iterations where the labeled pool grows, and a sequence of models that can be evaluated on the *Test* set are trained. Given a performance metric (e.g., recall at a fixed false positive rate), we obtain a learning curve where the metric usually improves during the simulation. Since we run 35 simulations, we obtain a distribution of learning curves, which we will visualize as percentile band plots in Section 4.

Since we run hundreds of experiments to test different policies, datasets and time periods, it is not feasible to observe all learning curves. Therefore we now define three aggregations to summarize each set of learning curves and be able to interpret the results.

Learning curves rise. To summarise how quickly the learning curves rise throughout the iterations (see, e.g., Figure 4), we compute the Area Under the percentile 50 learning curve (Area P50), defined as the curve tracing the median performance (over the 35 seeds) on each iteration. In addition, we normalize it by the area under the median optimistic baseline, which is the horizontal line corresponding to the median performance of the optimistic model (denoted by *Norm Area P50*). This allows us to compare folds relative to their optimistic baseline, while correcting for temporal drift unrelated to AL that also shifts that baseline.

Learning curves variability. To measure the variance of the learning curves (a good policy will always rise fast for all seeds), we use the Area between the percentiles 10 and 90 (denoted *Var* in the results). This is also normalized by the optimistic baseline area.

Quality of the final AL model. This is defined as the median performance of the final AL model normalized by the performance of the optimistic baseline (we denote it as *Norm Final P50*).

4 RESULTS

We now present results of the AL experiments for the various datasets focusing first on the most imbalanced. In Table 3, we show a summary of metrics for the five folds and all policies for Bank 1. Each row displays values for a specific policy sequence (1-stage, with only a cold policy, and 2-stage, without warm-up – see dashed lines). We have five groups of columns (one per *Fold*) with three metrics each (see Section 3.3.4): i) the normalized area under percentile 50 (Norm Area P50, blue density scale), ii) the ranking of the policy for the fold according to Norm Area P50 (center), and iii) the percentile 50 of the final normalized AL model performance (Norm Final P50, green density scale).

The rightmost pair of columns in Table 3 contains two metrics that summarize the five folds, namely the average of the ranks of each fold for each sequence (AVG Rank, orange density scale) and the average of the normalized area between percentiles 10 and 90 (AVG Var, red density scale). The former provides an overall measure of how fast the policy performance rises, whereas the latter of how noisy the policy is, for this dataset. The table rows are sorted by ascending AVG Rank. Therefore policies that perform better on various folds are at the top. We choose to rank by Norm Area P50 rather than Norm Final P50 because it is more sensitive to how quickly the learning curves rise, which is critical in systems that need a good model to start acting as early as possible. Nevertheless, the final model performance is important to tells us how close we get to the optimistic baseline. We include 12 sequences specified on the left. *Random* and *QueryAll* are baselines (Section 2.2.4).

Bank 1 is the most challenging dataset with an extremely large class imbalance. Therefore we doubled the daily review budget and trained in the full two weeks available for AL in the *Train* period (see weeks 3 and 4 of each *Fold* in Figure 3). The best policies in Table 3 outperform *Random* by a large margin (close to doubling the performance in some cases). Furthermore, they are on par with the *QueryAll* on folds 1, 4 and 5, both for the *Area* metric and the *Final* performance. In folds 2 and 3, although *QueryAll* performs substantially better, the group of top performing AL policies, based on uncertainty sampling, continue to rank highly.

Observe that, except for the rank, all the metrics have been normalized by the optimistic baseline, which is trained on extra data (full 4 weeks of the train period vs 2 weeks in Figure 3) with supervised feature selection and hyper-parameter tuning. This additional data would not be available in a realistic production setting and the improved training is challenging for AL in streaming. This explains why most metrics are smaller than 1. The exception is *Fold 5*, where Norm Final P50 is larger than 1 for various policies. This can be explained by observing the learning curves for *Fold 5* in Figure 4, where we show the distribution of learning curves for the best AL policy (left) and the *Random* policy (right)– represented by the rising green bands. Three equally spaced percentile bands are included, together with a solid gray line that traces the median. The distribution of values for the optimistic baseline is represented in the horizontal gray bands. All values have been normalized by the percentile 50 of the optimistic baseline. In this fold we can see that the distribution of values for the training of the optimistic baseline is quite wide. Thus, despite being above 1, the final performance of the AL model for the best policy is still within the central part of the distribution. Comparing left and right, we confirm that the 3-stages policy rises quick to high performance with a narrow variance.

It is also important to note that 3-stage sequences, i.e., with ODAL *warm-up*, tend to outperform the corresponding setups without ODAL, especially when paired with uncertainty based policies.

The overall conclusions, up to data set specific noise and some temporal drift effects, are confirmed for the other datasets. Note that AL typically only uses 2% to 10% of the number of samples available to the optimistic baselines. For other datasets we only present the policy rankings in Section 4.1, due to space constraints.

4.1 Aggregation over Datasets

In the previous section we discussed policy rankings and a pattern emerged: 3-stage sequences were the best performing policies, some 2-stage sequences also showed a good performance, and the rankings of the least performing policies were unstable across folds.

A convenient way of aggregating this information, to provide a clearer picture of the overall rankings, is to average out the policy ranks over the studied datasets. This is displayed in Table 4. As expected, overall, the *QueryAll* policy ranks first, even though it is not always the top one for some datasets. The 3-stage policies based on entropy or epistemic uncertainty rank very close to it, which indicates that these are high quality AL policies. Regarding sequences with *Expected Model Change* or the fraud percentile based *Uncertainty* policy, despite ranking in the middle of the table, for some datasets they rank very low, so they are not very stable/consistent. On the other hand, the 2-stage policy with ODAL ranks between 5 and 7 across datasets, which reinforces its value as a stable *warm-up* policy. The *Random* policy ranks low, as expected. *QBC* also ranks low, but this may be due to our specific/simple choice of committee (a more detailed study is left to future work). Another important observation is that all 3-stage policies rank higher than their 2-stages counterpart. In Figure 5 we display a visualization that helps understanding this improvement for the entropy based uncertainty policy. On each row we present the average increase of sampled positives, over all folds, when adding ODAL as a warm-up policy. For each fold, the increase is the 10th percentile difference between the positives obtained with a 3-stage sequence and the corresponding 2-stage sequence, divided by the mean positives of the 2-stage sequence. We can clearly observe that, for datasets with larger imbalances, including ODAL lifts up this low percentile considerably in early iterations (e.g., $\sim 3\times$ the mean value for *Bank 1*). The effect progressively disappears for milder imbalances – *Merchant*.

5 RELATED WORK

Various AL methods have been proposed and surveyed in the literature in the last decades, [18, 25]. In our experiments we focused on the *cold start* scenario with no historical data in a streaming

			Fold 1			Fold 2			Fold 3			Fold 4			Fold 5				
Cold	*Warmup*	*Hot*	Norm Area P50	Rank	Norm Final P50	Norm Area P50	Rank	Norm Final P50	Norm Area P50	Rank	Norm Final P50	Norm Area P50	Rank	Norm Final P50	Norm Area P50	Rank	Norm Final P50	AVG Rank	AVG Var
QueryAll	---	---	0.55	1	0.52	0.72	1	0.81	0.72	1	1.00	0.53	1	0.93	0.83	1	1.13	1.0	0.23
Random	ODAL	Unc. (entropy)	0.43	2	0.53	0.51	2	0.58	0.50	3	0.68	0.52	3	0.89	0.69	4	1.13	2.8	0.33
Random	ODAL	Unc. (epistemic)	0.43	4	0.51	0.49	4	0.62	0.50	4	0.79	0.52	2	0.93	0.72	3	1.20	3.4	0.34
Random	ODAL	Unc. (percentile)	0.43	3	0.60	0.51	3	0.54	0.48	5	0.64	0.45	6	0.81	0.69	5	1.07	4.4	0.32
Random	---	Unc. (entropy)	0.39	5	0.52	0.45	8	0.69	0.48	6	0.68	0.47	4	0.89	0.74	2	1.20	5.0	0.40
Random	---	ODAL	0.30	7	0.52	0.44	9	0.50	0.55	2	0.89	0.27	11	0.30	0.61	7	1.13	7.2	0.23
OutlierDetect	---	---	0.26	8	0.51	0.49	5	0.64	0.40	9	0.57	0.33	9	0.37	0.57	9	1.00	8.0	0.25
Random	ODAL	EMC	0.10	12	0.07	0.48	6	0.65	0.42	8	0.50	0.30	10	0.22	0.66	6	1.13	8.4	0.19
Random	---	QBC	0.16	11	0.27	0.40	10	0.62	0.43	7	0.64	0.46	5	0.70	0.55	11	0.87	8.8	0.35
Random	ODAL	QBC	0.17	10	0.30	0.46	7	0.58	0.39	10	0.61	0.42	7	0.59	0.56	10	0.93	8.8	0.29
Random	---	---	0.36	6	0.56	0.40	11	0.58	0.37	12	0.50	0.34	8	0.48	0.54	12	0.80	9.8	0.38
Random	---	EMC	0.17	9	0.37	0.37	12	0.54	0.39	11	0.50	0.12	12	0.37	0.61	8	0.93	10.4	0.40

Table 3: *Bank 1 rankings of AL policies using various folds* (see also detailed description in the text).

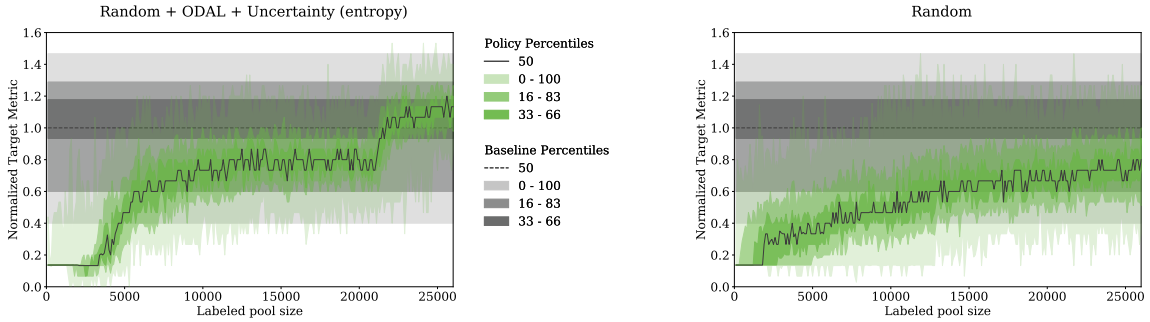

Figure 4: *Learning curves distribution for Bank 1 in the best fold (5)*: The best sequence of policies (left panel green bands), and the *Random* policy (right panel green band), normalized by the percentile 50 of the optimistic baseline (gray bands).

Cold	*Warmup*	*Hot*	Bank 1	Bank 2	Payment Processor	Merchant	AVG
QueryAll	---	---	1.0	4.4	3.0	2.8	2.8
Random	ODAL	Unc. (entropy)	2.8	2.6	3.0	3.2	2.9
Random	ODAL	Unc. (epistemic)	3.4	1.6	4.8	3.4	3.3
Random	---	Unc. (entropy)	5.0	3.8	4.4	2.8	4.0
Random	---	ODAL	7.2	5.0	7.4	7.0	6.7
Random	ODAL	EMC	8.4	9.2	4.4	5.4	6.9
Random	---	EMC	10.4	10.4	4.2	5.4	7.6
Random	ODAL	Unc. (percentile)	4.4	6.2	10.8	9.4	7.7
OutlierDetect	---	---	8.0	7.2	7.6	8.6	7.9
Random	---	---	9.8	10.8	8.2	8.2	9.3
Random	ODAL	QBC	8.8	8.0	10.4	10.6	9.5
Random	---	QBC	8.8	8.8	9.8	10.8	9.6

Table 4: *Overall policy ranking*: Average ranks for each dataset (four central columns) and their overall average (right column). Rows are sorted by the AVG column.

environment where the unlabeled data pool grows. This is in contrast with typical AL setups where the data source is static. Some studies have appeared in the literature discussing AL methods in a streaming data scenario [2, 3, 8, 9, 13, 22, 26–29]. Notably, Carcillo et al. [2], investigated several AL methods for a credit card fraud dataset. Instances were selected once a day with AL, according to a fixed budget. In contrast, we consider scenarios where several small batches of instances are processed during the day to exploit the collected labels more frequently to update the AL policy, which is important to avoid the selection of many similar instances in one large batch. Furthermore, we presented a detailed analysis of AL

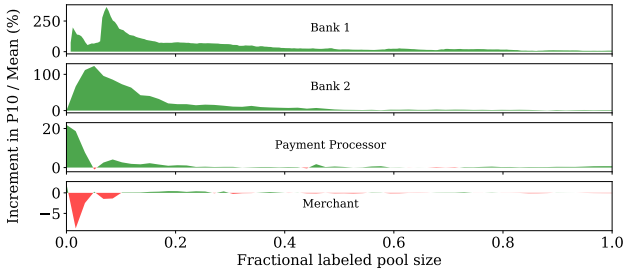

Figure 5: *Boost in the number of positives sampled in 3-stages vs 2-stages for the entropy based uncertainty policy* (see detailed description in the text).

curves in the fraud domain, to provide a more complete understanding of its effectiveness for fraud detection, as well as investigated new policies that were not considered in Carcillo et al. [2]. Finally, in reference [2] no analysis of AL curves was presented, nor of their variability, which is essential to observe the boost in ML model performance at early stages of the AL process. Other studies in our literature review, cited above, are either: i) focused on applying AL to address concept drift, or ii) not focused on highly imbalanced problems, or iii) not focused on dealing with the *cold start* problem.

6 CONCLUSIONS

We studied the problem of creating a small labeled dataset, with a limited budget of annotations by analysts, in a streaming environment, in a cold start scenario (no previously labeled data and little

or no unlabeled data) for highly imbalanced datasets. We proposed an AL system adapted to these conditions and performed a detailed study on four real world credit card fraud detection datasets, covering three use cases with several orders of magnitude in class imbalances. We proposed various ingredients that proved essential, namely: i) ODAL, a computationally efficient version of discriminative active learning to quickly represent well the unlabeled pool in the labeled pool, relying only on the labeled pool features distribution, and ii) the combination of ODAL, as a warmup-policy, with other AL polices, in a 3-stage sequence to alleviate the cold start problem in highly imbalanced datasets where it may take a long time until some of the labels are found. We also proposed two alternative uncertainty measures for the *Uncertainty Sampling* policy – epistemic uncertainty and the fraud percentile measure – as well as an alternative measure of disagreement based on rank differences for *Query By Committee*.

In Section 3 we conducted detailed experimental studies, including optimistic baselines and 12 different policy sequences to be ranked. Our analysis showed that the best performing AL policies are 3-stage sequences with ODAL warm-up and Uncertainty Sampling as Hot policy (either entropy or epistemic). In particular, we showed that the ODAL warm-up boosts the learning curves in the earlier AL iterations. As a general rule, the final overall ranking shows that including ODAL warm-up before any *Hot* policy boosts its learning curves, especially for large class imbalance. Furthermore, the best performing sequence is often as good as the *QueryAll* policy, it has low variance learning curves, it is competitive with the optimistic baseline and substantially better than *Random*. Our results show that the required amount of labeled examples, until the learning curve stabilizes, often ranges between 3 000 to 6 000 for mild to intermediate class imbalances, and a bit over 20 000 for extreme imbalances ($\sim$ 2% to 10% of the optimistic baseline data).

To conclude, we comment on some future directions. In this study, we have simulated the analyst queries by using the real labels in the datasets. It would be interesting to perform experiments with real analysts in a live environment to see if the performance gains are confirmed. Finally, we have not touched upon other possible problems and improvements that could be important in a real system. This includes the issue of evaluating the AL models online – in our study we used an independent test set in the future of the train set for evaluation. Related to this, it would also be interesting to include online hyper-parameter tuning and model selection, as well as online supervised feature selection, instead of using a static set of features selected in an unsupervised way on the first day.

REFERENCES

[1] Bernardo Branco, Pedro Abreu, Ana Sofia Gomes, Mariana S. C. Almeida, João Tiago Ascensão, and Pedro Bizarro. 2020. Interleaved Sequence RNNs for Fraud Detection. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 3101–3109. https://doi.org/10.1145/3394486.3403361

[2] Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, and Gianluca Bontempi. 2018. Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization. *International Journal of Data Science and Analytics* 5 (2018), 285–300.

[3] Yu Cheng, Zhengzhang Chen, Lu Liu, Jiang Wang, Ankit Agrawal, and Alok Choudhary. 2013. Feedback-Driven Multiclass Active Learning for Data Streams. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management* (San Francisco, California, USA) *(CIKM '13)*. Association for Computing Machinery, New York, NY, USA, 1311–1320. https://doi.org/10.1145/2505515.2505528

[4] David A. Cohn. 1996. Neural Network Exploration Using Optimal Experiment Design. *Neural Networks* 9, 6 (1996), 1071 – 1083. https://doi.org/10.1016/0893-6080(95)00137-9

[5] Atsushi Fujii, Kentaro Inui, Takenobu Tokunaga, and Hozumi Tanaka. 1998. Selective Sampling for Example-based Word Sense Disambiguation. *Computational Linguistics* 24, 4 (1998), 573–597. https://www.aclweb.org/anthology/J98-4002

[6] Daniel Gissin and Shai Shalev-Shwartz. 2019. Discriminative Active Learning. *CoRR* abs/1907.06347 (2019). arXiv:1907.06347 http://arxiv.org/abs/1907.06347

[7] Neil Houlsby, Jose Miguel Hernandez-Lobato, and Zoubin Ghahramani. 2014. Cold-start Active Learning with Robust Ordinal Matrix Factorization. In *Proceedings of the 31st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 32)*, Eric P. Xing and Tony Jebara (Eds.). PMLR, Bejing, China, 766–774. http://proceedings.mlr.press/v32/houlsby14.html

[8] Janardan and Shikha Mehta. 2017. Concept drift in Streaming Data Classification: Algorithms, Platforms and Issues. *Procedia Computer Science* 122 (2017), 804 – 811. https://doi.org/10.1016/j.procs.2017.11.440 5th International Conference on Information Technology and Quantitative Management, ITQM 2017.

[9] Janez Kranjc, Jasmina Smailović, Vid Podpečan, Miha Grčar, Martin Žnidaršič, and Nada Lavrač. 2015. Active learning for sentiment analysis on data streams: Methodology and workflow implementation in the ClowdFlows platform. *Information Processing and Management* 51, 2 (2015), 187 – 203. https://doi.org/10.1016/j.ipm.2014.04.001

[10] David D. Lewis and William A. Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Dublin, Ireland) *(SIGIR '94)*. Springer-Verlag, Berlin, Heidelberg, 3–12.

[11] F. T. Liu, K. M. Ting, and Z. Zhou. 2008. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*. 413–422.

[12] Paulo Marques, Miguel Araújo, Bruno Laraña, Nuno Diegues, Pedro Silva, and Pedro Bizarro. 2019. Semantic-aware feature engineering, US20200090003A1 Patent (Pending).

[13] H. Nguyen, J. B. Gomes, M. Wu, H. Cao, J. Cao, and S. Krishnaswamy. 2015. Active learning for accurate analysis of streaming partial discharge data. In *2015 IEEE Conference on Prognostics and Health Management (PHM)*. 1–5. https://doi.org/10.1109/ICPHM.2015.7245026

[14] Hieu T. Nguyen and Arnold Smeulders. 2004. Active Learning Using Pre-Clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning* (Banff, Alberta, Canada) *(ICML '04)*. Association for Computing Machinery, New York, NY, USA, 79. https://doi.org/10.1145/1015330.1015349

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[16] A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi. 2015. Calibrating Probability with Undersampling for Unbalanced Classification. In *2015 IEEE Symposium Series on Computational Intelligence*. 159–166.

[17] Nicholas Roy and Andrew McCallum. 2001. Toward Optimal Active Learning through Sampling Estimation of Error Reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 441–448.

[18] Burr Settles. 2009. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison. http://axon.cs.byu.edu/~martinez/classes/778/Papers/settles.activelearning.pdf

[19] Burr Settles, Mark Craven, and Soumya Ray. 2008. Multiple-Instance Active Learning. In *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (Eds.). Curran Associates, Inc., 1289–1296. http://papers.nips.cc/paper/3252-multiple-instance-active-learning.pdf

[20] H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by Committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (Pittsburgh, Pennsylvania, USA) *(COLT '92)*. Association for Computing Machinery, New York, NY, USA, 287–294. https://doi.org/10.1145/130385.130417

[21] Mohammad Hossein Shaker and Eyke Hüllermeier. 2020. Aleatoric and Epistemic Uncertainty with Random Forests. In *Advances in Intelligent Data Analysis XVIII*, Michael R. Berthold, Ad Feelders, and Georg Krempl (Eds.). Springer International Publishing, Cham, 444–456.

[22] J. Shan, H. Zhang, W. Liu, and Q. Liu. 2019. Online Active Learning Ensemble Framework for Drifted Data Streams. *IEEE Transactions on Neural Networks and Learning Systems* 30, 2 (2019), 486–498. https://doi.org/10.1109/TNNLS.2018.2844332

[23] Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2, 1-3 (1987), 37–52.

[24] Zuobing Xu, Ram Akella, and Yi Zhang. 2007. Incorporating Diversity and Density in Active Learning for Relevance Feedback. In *Advances in Information Retrieval*, Giambattista Amati, Claudio Carpineto, and Giovanni Romano (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 246–257.

[25] Yazhou Yang and Marco Loog. 2018. A benchmark and comparison of active learning for logistic regression. *Pattern Recognition* 83 (2018), 401 – 415. https://doi.org/10.1016/j.patcog.2018.06.004

[26] Zhilin Yang, Jie Tang, and Yutao Zhang. 2014. Active Learning for Streaming Networked Data. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (Shanghai, China) *(CIKM '14)*. Association for Computing Machinery, New York, NY, USA, 1129–1138. https://doi.org/10.1145/2661829.2661981

[27] Y. Zhang, P. Zhao, S. Niu, Q. Wu, J. Cao, J. Huang, and M. Tan. 2019. Online Adaptive Asymmetric Active Learning with Limited Budgets. *IEEE Transactions on Knowledge and Data Engineering* (2019), 1–1.

[28] X. Zhu, P. Zhang, X. Lin, and Y. Shi. 2010. Active Learning From Stream Data Using Optimal Weight Classifier Ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 40, 6 (2010), 1607–1621. https://doi.org/10.1109/TSMCB.2010.2042445

[29] Indrė Žliobaitė, Albert Bifet, Bernhard Pfahringer, and Geoff Holmes. 2011. Active Learning with Evolving Streaming Data. In *Machine Learning and Knowledge Discovery in Databases*, Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 597–612.