Article

Not peer-reviewed version

scikit-activeml: A Comprehensive and User-friendly Active Learning Library

Marek Herde *, Minh Tuan Pham *, Daniel Kottke, Alexander Benz, Lukas Lührs, Pascal Mergard, Christoph Sandrock, Jiaying Cheng, Atal Roghman, Mehmet Müjde, Lukas Rauch, Bernhard Sick

Posted Date: 4 July 2025

doi: 10.20944/preprints202507.0252.v1

Keywords: active learning; query strategy; classification; regression; python

 doi:10.20944/preprints202507.0252.v1

Article

`scikit-activeml`: A Comprehensive and User-Friendly Active Learning Library

Marek Herde [1,*], Minh Tuan Pham [1,*], Daniel Kottke [2], Alexander Benz [1], Lukas Lührs [1], Pascal Mergard [1], Christoph Sandrock [3], Jiaying Cheng [1], Atal Roghman [1], Mehmet Müjde [1], Lukas Rauch [1] and Bernhard Sick [1]

[1] University of Kassel, Germany
[2] Deutsche Bahn AG, Germany
[3] TU Wien, Austria
* Correspondence: Equally contributing authors contactable via marek.herde@uni-kassel.de & tuan.pham@uni-kassel.de

Abstract

`scikit-activeml` is a user-friendly open-source Python library for active learning on top of `scikit-learn`. Included are implementations of a large collection of query strategies, models, and visualization tools in pool- and stream-based active learning for classification or regression tasks with single or multiple annotators. The flexible design of the active learning cycle enables individual adaptations to a variety of learning scenarios. Our source code with comprehensive documentation is available at https://scikit-activeml.github.io.

Keywords: active learning; query strategy; classification; regression; python

1. Introduction

Supervised machine learning models require labeled data to perform well. While unlabeled data can be easily gathered, the labeling process is difficult, time-consuming, or expensive in many applications. Active learning (Settles 2009) aims to solve this issue by querying labels for samples that maximize the model's performance. Therefore, many query strategies have been proposed, leading to broader applicability (Li et al. 2024; Cacciarelli and Kulahci 2024). Yet, the query strategies' varying requirements resulted in divergent, mostly incompatible implementations lacking thorough testing. A unified library can address these challenges by offering query strategies in a standardized structure with comprehensive documentation and visualization tools. Such a library supports reproducible large-scale evaluation studies, enabling straightforward comparisons with baseline and state-of-the-art query strategies.

`scikit-activeml` is a Python package for active learning, whose **contributions** are:

- implementations of 61 pool- and stream-based query strategies, drawn from about 40 research articles and covering classification and regression tasks,
- adopting the design patterns and structure from `scikit-learn` (Pedregosa et al. 2011), enabling the integration into existing pipelines,
- a modular and user-friendly structure allowing the development of query strategies and active learning cycles according to users' individual requirements,
- and extensive documentation with illustrative examples and tutorials, e.g, image/text classification, interactive labeling, and templates for experimental studies.

2. Structural Overview

Our active learning library `scikit-activeml` adopts the design principles, e.g., interfaces, docstrings, and nomenclature, of the popular machine learning framework `scikit-learn` (Pedregosa et al. 2011). Figure 1 details the corresponding overall structure.

 doi:10.20944/preprints202507.0252.v1

The core part of `scikit-activeml` is the `QueryStrategy` interface enforcing the implementation of a `query` method. This method is responsible for selecting data samples to be labeled. Since such a selection depends on the respective active learning paradigm, there are two subinterfaces differing between pool- and stream-based active learning. Query strategies for pool-based active learning (Li et al. 2024) assume the availability of a large pool of unlabeled samples from which samples can be selected. Accordingly, the `query` method takes this pool as input and outputs the selected samples. Optionally, this method can return a query strategy's estimated utility scores for querying labels of samples. In contrast, stream-based active learning (Cacciarelli and Kulahci 2024) assumes a stream of incoming data samples. Each incoming sample is assessed individually or in batches for deciding whether to label the sample(s). All stream-based query strategies implement an `update` method to track the history of queries and may be combined with an optional budget manager to control the number of label acquisitions over time. Another differentiation of active learning concerns the label source, typically distinguished between a single omniscient annotator and multiple error-prone annotators (Herde et al. 2021). In the latter case, annotators may provide incorrect labels, requiring the query strategy to guide both sample and annotator selection. Currently, `scikit-activeml` offers multi-annotator learning exclusively for pool-based active learning, reflecting where most research is concentrated.

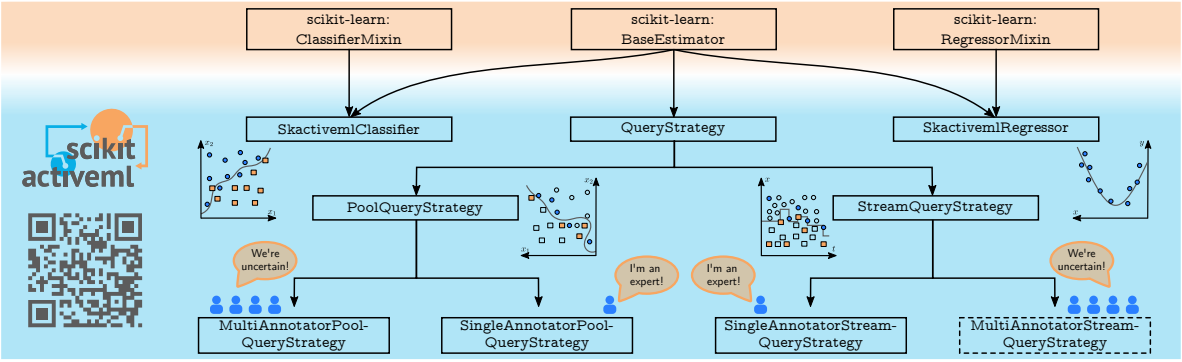

Figure 1. Nodes depict the core abstract classes of `scikit-activeml (v0.6)`, derived from `scikit-learn` abstract classes, arrows inheritance and dashed nodes not yet supported features. `scikit-learn` is used here for identification only and does not imply endorsement.

Beyond query strategies, machine learning models represent another core part of `scikit-activeml`. So far, classification and regression models are supported. All of them either implement the `ClassifierMixin` or `RegressorMixin` interface to ensure compatibility with `scikit-learn`. The main difference to `scikit-learn` models is that our extension natively accepts unlabeled and labeled samples as training input. These `scikit-learn` models are integrated into our library with easy-to-use wrappers providing functionalities, such as cost-sensitive classification and probability density functions as regression outputs.

An active learning cycle is defined through a query strategy and a machine learning model. Algorithm 1 demonstrates this in a pool-based scenario with a single annotator. Unlabeled data `X` is categorized into `centers=8` classes. Current labels are stored in `y`, where unlabeled samples are indicated as `missing_label=-1`. As an example, ten random samples are labeled at the start. Using a logistic regression model wrapped with `SklearnClassifier` for filtering unlabeled samples in `X` and `y`, the cycle employs *batch active learning by diverse gradient embeddings* (BADGE, Ash et al., 2020) as query strategy returning the indices `q_ids` of the `batch_size=3` selected samples, whose labels in `y` are replaced with their true values from `y_true`. The querying continues until the maximum cycles are reached. Appendix A and our documentation provide code snippets of further active learning scenarios.

 doi:10.20944/preprints202507.0252.v1

Algorithm 1: Code snippet of a pool-based active learning cycle for classification.

```python
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_blobs
from skactiveml.pool import Badge
from skactiveml.classifier import SklearnClassifier

# Generate data set with 10 initially labeled samples.
X, y_true = make_blobs(n_samples=500, centers=8)
y = np.full_like(y_true, fill_value=-1)
rand_ids = np.random.choice(len(X), size=10)
y[rand_ids] = y_true[rand_ids]

# Create classifier and query strategy.
clf = SklearnClassifier(
    LogisticRegression(max_iter=1000),
    classes=np.unique(y_true),
    missing_label=-1,
)
qs = Badge(missing_label=-1)

# Execute active learning cycle.
n_cycles = 20
for c in range(n_cycles):
    q_ids = qs.query(X=X, y=y, clf=clf, batch_size=3)
    y[q_ids] = y_true[q_ids]

# Fit final classifier.
clf.fit(X, y)
```

Despite `scikit-activeml`'s focus on classic `scikit-learn` machine learning models for tabular data, we also support deep learning models for image, text, and audio data together with state-of-the-art query strategies for deep active learning. For this purpose, users may implement their own architectures in `pytorch` (Paszke et al. 2019) to be wrapped by `skorch` (Tietz et al. 2017) or they may employ pre-trained deep learning models, e.g., DINOv2 (Oquab et al. 2023), BERT (Devlin et al. 2019), and Wav2Vec (Baevski et al. 2020), as feature extractors. The latter approach is increasingly popular due to pre-trained deep learning models' ability to infer rich, task-agnostic embeddings enhancing sample selection efficiency and performance in label-scarce scenarios (Hacohen et al. 2022; Yehuda et al. 2022; Huseljic et al. 2024; Gupte et al. 2024).

3. Related Work

Next to `scikit-activeml`, there exist several open-source Python projects for active learning. Table 1 provides an overview regarding the active learning scenarios supported by the individual projects, while Table 2 informs about the organization of the projects' public repositories. Appendix C details the definitions of the tables' attributes.

A key difference between `scikit-activeml` and other active learning projects is its comprehensiveness indicated by a large number of query strategies (cf. Appendix B) covering multiple active learning scenarios, which aligns with the objective of being a library without focusing on a specific data type or query strategies. The most related active learning project is `modal`, lastly released in 2023,

 doi:10.20944/preprints202507.0252.v1

which also supports many active learning scenarios with `scikit-learn` as the main machine learning framework. However, `modal` encapsulates fitting, querying, and labeling all in one class, which is straightforward but less flexible than `scikit-activeml` separating query strategies from the models. Since 2024, only four active learning projects, including `scikit-activeml`, have published new releases. Among them, `baal` focuses on Bayesian techniques, while `small-text` specifically targets text data. With a 100% code line coverage, `scikit-activeml` aims at providing correct implementations.

Table 1. Feature comparison between active learning projects (as of April 29, 2025).

General		Query Strategies				Learning Tasks	
Name	Authors	Number	Pool	Stream	Multi-annotator	Regression	Classification
`DeepAL`	Huang	9	✓	✗	✗	✗	✓
`libact`	Yang et al.	19	✓	✗	✓	✗	✓
`alipy`	Tang et al.	30	✓	✗	✓	✗	✓
`modal`	Danka and Horvath	21	✓	✓	✗	✓	✓
`altoolbox`	Tsvigun et al.	19	✓	✗	✗	✗	✓
`baal`	Atighehchian et al.	10	✓	✓	✗	✓	✓
`pyrelational`	Scherer et al.	17	✓	✗	✗	✓	✓
`small-text`	Schröder et al.	23	✓	✗	✗	✗	✓
`scikit-activeml`	Herde et al.	61	✓	✓	✓	✓	✓

Table 2. Repository comparison between active learning projects (as of April 29, 2025).

Name	Implementation					Documentation	
	Last Release	Frameworks	License	Tests	Coverage	Tutorials	Developer Guide
`DeepAL`	N/A	`pytorch`	MIT	✗	N/A	✗	✗
`libact`	2019	`scikit-learn`	BSD-2-Clause	✓	90 %	✓	✓
`alipy`	2021	`scikit-learn`	BSD-3-Clause	✓	N/A	✓	✗
`modal`	2023	`scikit-learn`, `keras`, `skorch`	MIT	✓	97 %	✓	✓
`altoolbox`	2022	`pytorch`, `transformers`	MIT	✓	N/A	✗	✓
`baal`	2024	`pytorch`, `transformers`	Apache 2.0	✓	N/A	✓	✓
`pyrelational`	2024	`scikit-learn`, `pytorch(-lightning)`	Apache 2.0	✓	94 %	✓	✓
`small-text`	2024	`scikit-learn`, `pytorch`, `transformers`	MIT	✓	96 %	✓	✓
`scikit-activeml`	2025	`scikit-learn`, `skorch`, `river`[1]	BSD-3-Clause	✓	100 %	✓	✓

4. Conclusion

We briefly outlined the main concepts of `scikit-activeml` and compared them to related Python active learning projects. Our ongoing objective is to unify active learning processes for broader adoption. Therefore, we plan to incorporate more active learning scenarios, e.g., multi-label classification (Wu et al. 2020), multi-output regression (Li et al. 2022), and stopping criteria (Vlachos 2008). A major limitation of `scikit-activeml` is that it does not natively support highly complex deep learning tasks, e.g., object detection, but focuses on `scikit-learn` workflows to keep the library simple and serve as a code reference for adapting its query strategies to more specialized active learning projects.

Acknowledgments: This project was funded by the state of Hesse at the University of Kassel in Germany.

[1] `scikit-activeml (v0.6)` depends on `scikit-learn (v1.6)`, which is unsupported by the latest release of `river (v0.22)`. Therefore, `river (v0.22)` is only compatible with `scikit-activeml (v0.5)`.

 doi:10.20944/preprints202507.0252.v1

Appendix A. Beyond Pool-Based Active Learning for Classification

Beyond pool-based active learning for classification with an omniscient annotator, this section provides code snippets for regression tasks, stream-based active learning, and active learning with multiple error-prone annotators.

Algorithm 2: Code snippet of a pool-based active learning cycle for regression.

```
import numpy as np
from sklearn.gaussian_process import GaussianProcessRegressor
from skactiveml.pool import GreedySamplingTarget
from skactiveml.regressor import SklearnRegressor

# Generate data set.
random_state = np.random.RandomState(0)
X = random_state.uniform(size=(200, 1), low=-1, high=1)
y_true = np.sin(X.ravel()) + random_state.randn(len(X)) * 0.1
y = np.full_like(y_true, fill_value=np.nan)

# Use 10 samples as initial training data.
y[:10] = y_true[:10]

# Create regressor and query strategy.
gp = GaussianProcessRegressor(random_state=0)
reg = SklearnRegressor(gp, random_state=0, missing_label=np.nan)
qs = GreedySamplingTarget(method="GSi", random_state=0)

# Execute active learning cycle.
n_cycles = 5
for c in range(n_cycles):
    reg.fit(X, y)
    q_ids, u_scores = qs.query(X=X, y=y, reg=reg, fit_reg=False, batch_size=10, return_utilities=
        True)
    y[q_ids] = y_true[q_ids]

# Fit final regressor.
reg.fit(X, y)
```

Algorithm 2 shows the code snippet of a simple example of **pool-based active learning for regression** (Kumar and Gupta 2020) with a single annotator. Thereby, the main difference to the code snippet for classification in Algorithm 1 is the usage of Gaussian processes as a regression model. Further, *improved greedy sampling* (iGS, Wu et al., 2019) is employed as a dedicated query strategy for regression tasks. The training labels `y` contain the target values for all observed samples `X`, where the symbol `np.nan` marks unlabeled samples, i.e., the ones where the target values have not been acquired yet. In each active learning cycle, `batch_size=10` of the unlabeled samples are selected for label acquisitions by overriding the `np.nan` symbol with the actual target values. In this example, the regressor is fitted manually outside of the `query` method by the user as an option for improved flexibility, which can be useful if the fitting process requires specialized handling. Moreover, the utilities `u_scores` computed by the query strategy are returned for illustrative purposes.

Algorithm 3 shows a code snippet of a simple example of **pool-based active learning with multiple annotators** (Herde et al. 2021; Donmez et al. 2009) for classification. In this scenario, we can only access error-prone annotators. Therefore, we aim to not only select the most informative samples but also annotators who provide correct labels for these samples. For this purpose, the most

 doi:10.20944/preprints202507.0252.v1

straightforward approach is to employ a common query strategy for selecting samples and then using multi-annotator learning models to guide the selection of annotators. These models estimate annotators' performances (Raykar et al. 2010; Herde et al. 2024) to correct their noisy labels for training accurate classifiers. In our code snippet, `y_noisy` represents the class labels of four annotators, where the first three columns correspond to three omniscient annotators and the last column to one adversarial annotator. We train a logistic regression model via the expectation-maximization algorithm with the true labels as latent variables (Raykar et al. 2010). As a result, we obtain a logistic regression model for selecting samples via *typical clustering* (TypiClust, Hacohen et al., 2022) and annotator-wise confusion matrices to estimate `A_perf` as their performances per sample for selecting the best annotators. For this purpose, different selection schemes are possible by varying the number of annotators to be queried per sample. In our example, we select `batch_size=3` samples per active learning cycle, and only a single, i.e., the estimated best, annotator for each of these three selected samples (`n_annotators_per_sample=1`).

Algorithm 3: Code snippet of a pool-based active learning cycle with noisy annotators.

```python
import numpy as np
from sklearn.datasets import make_blobs
from skactiveml.pool import TypiClust
from skactiveml.pool.multiannotator import SingleAnnotatorWrapper
from skactiveml.classifier.multiannotator import AnnotatorLogisticRegression

# Generate data set with four annotators, of which the last one is always wrong.
X, y_true = make_blobs(n_samples=200, centers=5, random_state=0)
classes = np.unique(y_true)
y_noisy = np.column_stack((y_true, y_true, y_true, (y_true + 1) % 5))

# Perform active learning with zero initial labels.
y = np.full(shape=y_noisy.shape, fill_value=-1)

# Create classifier and query strategy.
clf = AnnotatorLogisticRegression(missing_label=-1, classes=classes, random_state=0)
qs = TypiClust(missing_label=-1, random_state=0)
ma_qs = SingleAnnotatorWrapper(qs, random_state=0, missing_label=-1)

# Function to be able to index via an array of indices.
idx = lambda A: (A[:, 0], A[:, 1])

# Execute active learning cycles.
n_cycles = 50
for c in range(n_cycles):
    # Fit multi-annotator classifier and compute annotators' accuracies per sample.
    clf.fit(X, y)
    A_perf = clf.predict_annotator_perf(X)

    # Select samples and acquire labels.
    q_ids = ma_qs.query(X=X, y=y, batch_size=3, A_perf=A_perf, n_annotators_per_sample=1)
    y[idx(q_ids)] = y_noisy[idx(q_ids)]

# Fit final multi-annotator classifier.
clf.fit(X, y)
```

 doi:10.20944/preprints202507.0252.v1

Algorithm 4 shows a code snippet of a simple example of **stream-based active learning** (Cacciarelli and Kulahci 2024) with a single annotator. The data stream is given as unlabeled samples `X` and their labels `y_true`. The training data is managed with `X_train` and `y_train`. Here, we use a Parzen window classifier and Split (Žliobaitė et al. 2014) as the query strategy. The budget is set to 0.1 by default, i.e., the query strategy is allowed to query the labels of 10% of the incoming unlabeled samples. The budget can be adjusted via the `budget` attribute for each stream-based query strategy. The classifier is retrained for each new sample. Contrary to the pool scenario, many stream-based query strategies need to keep track of the budget. Thus, after querying whether to label a sample or not, the available budget for the query strategy is updated. The separation between querying labels and updating the available budget allows query strategies to be used within other strategies where the decision may be altered by a wrapper strategy.

Algorithm 4: Code snippet of a stream-based active learning cycle for classification.

```
import numpy as np
from sklearn.datasets import make_blobs
from skactiveml.classifier import ParzenWindowClassifier
from skactiveml.stream import Split
from skactiveml.utils import MISSING_LABEL

# Generate data set.
X, y_true = make_blobs(n_samples=200, centers=4, random_state=0)

# Initializing the training data as an empty array.
X_train, y_train = [], []

# Create classifier and query strategy.
clf = ParzenWindowClassifier(random_state=0, classes=np.unique(y_true))
qs = Split(random_state=0)

# Execute active learning cycle.
for x_t, y_t in zip(X, y_true):
    X_cand = x_t.reshape([1, -1])
    y_cand = y_t
    clf.fit(X_train, y_train)
    queried_indices = qs.query(candidates=X_cand, clf=clf)
    qs.update(candidates=X_cand, queried_indices=queried_indices)
    X_train.append(x_t)
    y_train.append(y_cand if len(queried_indices) > 0 else MISSING_LABEL)
```

Appendix B. Overview of Query Strategies

Table A1 overviews the 47 pool-based and Table A2 the 14 stream-based query strategies implemented by `scikit-activeml`. These strategies originate from over 40 research articles, some of which introduce multiple variants. For example, uncertainty sampling can be used with different uncertainty measures (Settles 2009). We indicate each strategy's target learning tasks (regression and/or classification), flag multi-annotator scenarios, and mark strategies considering diversity between samples within a selected batch. Furthermore, we categorize query strategies by their selection principles, i.e., informativeness (model uncertainty), representativeness (data-distribution coverage), and hybrid (combining both). Figure A1 provides a mind map that illustrates these different attributes of a query strategy.

 doi:10.20944/preprints202507.0252.v1

Table A1. Pool-based query strategies implemented by `scikit-activeml` (part I).

Name	Classi-fication	Regres-sion	Multi-annotator	Diverse Batches	Variants	References
Baselines and Wrappers						
Random Sampling	✓	✓	✗	✓	1	N/A
Subsampling Wrapper	✓	✓	✗	N/A	1	N/A
Parallel Selection Wrapper	✓	✓	✗	N/A	1	N/A
Single Annotator Wrapper	✓	✓	✓	N/A	1	N/A
Informativeness						
Uncertainty Sampling (US)	✓	✗	✗	✗	3	Lewis and Gale (1994) Scheffer et al. (2001) Settles (2009)
Query by Committee (QbC)	✓	✓	✗	✗	4	Seung et al. (1992) Engelson and Dagan (1996) McCallum and Nigamy (1998) Burbidge et al. (2007) Beluch et al. (2018)
Expected Model Variance Reduction	✗	✓	✗	✗	1	Cohn et al. (1996)
Expected Error Reduction (EER)	✓	✓	✗	✗	2	Roy and McCallum (2001)
Value of Information (VOI)	✓	✓	✗	✗	3	Margineantu (2005) Kapoor et al. (2007) Joshi et al. (2009)
Interval Estimation Threshold (IEThresh)	✓	✗	✓	✗	1	Donmez et al. (2009)
Bayesian Active Learning by Disagreement (BALD)	✓	✗	✗	✗	1	Houlsby et al. (2011)
Expected Model Change (EMC)	✗	✓	✗	✗	1	Cai et al. (2013)
Active Learning with Cost Embedding (ALCE)	✓	✗	✗	✗	1	Huang and Lin (2016)
Uncertainty Sampling via Maximizing Average Precision (USAP)	✓	✗	✗	✗	1	Wang et al. (2018)
Expected Model Output Change (EMOC)	✗	✓	✗	✗	1	Käding et al. (2018)
Cost-sensitive Uncertainty Sampling (CsUS)	✓	✗	✗	✗	1	Chen and Lin (2013)
Batch Bayesian Active Learning by Disagreement (BatchBALD)	✓	✗	✗	✓	1	Kirsch et al. (2019)
Epistemic Uncertainty Sampling (EpisUS)	✓	✗	✗	✗	1	Nguyen et al. (2019)
Regression-based Kullback-Leibler Divergence Maximization	✗	✓	✗	✗	1	Elreedy et al. (2019)
Contrastive Active Learning (CAL)	✓	✗	✗	✗	1	Margatina et al. (2021)

Continued on the next page.

 doi:10.20944/preprints202507.0252.v1

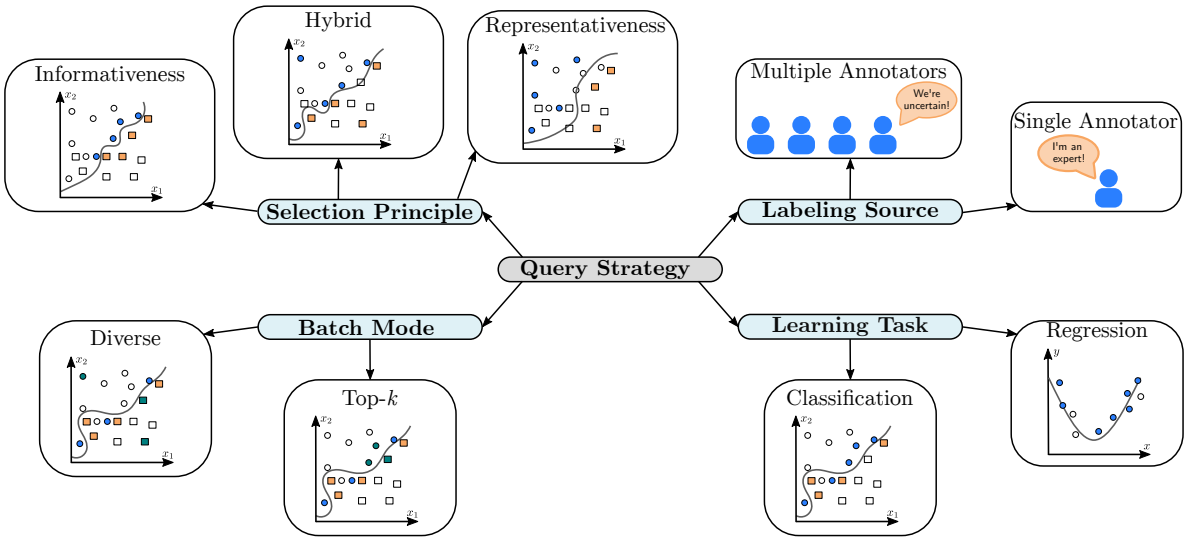

Figure A1. Mind map for categorizing the query strategies implemented by `scikit-activeml`.

Table A2. Pool-based query strategies implemented by `scikit-activeml` (part II).

Name	Classi-fication	Regres-sion	Multi-annotator	Diverse Batches	Variants	References
Representativeness						
Core Set	✓	✗	✗	✓	1	Sener and Savarese (2018)
Discriminative Active Learning (DAL)	✓	✓	✗	✓	1	Gissin and Shalev-Shwartz (2019)
Greedy Sampling (GS)	✓	✓	✗	✓	3	Wu et al. (2019)
Typicality Clustering (TypiClust)	✓	✓	✗	✓	1	Hacohen et al. (2022)
Probability Coverage (ProbCover)	✓	✗	✗	✓	1	Yehuda et al. (2022)
Regression Tree Active Learning (RT-AL)	✗	✓	✗	✓	3	Jose et al. (2023)
Hybrid						
Density-weighted Uncertainty Sampling (DwUS)	✓	✗	✗	✗	1	Donmez et al. (2007)
Dual Strategy for Active Learning (DUAL)	✓	✗	✗	✗	1	Donmez et al. (2007)
Querying Informative and Representative Examples (QUIRE)	✓	✗	✗	✗	1	Huang et al. (2010) Huang et al. (2014)
Density-Diversity-Distribution -Distance Sampling (4DS)	✓	✗	✗	✓	1	Reitmaier and Sick (2013)
Multi-class Probabilistic Active Learning (McPAL)	✓	✗	✗	✗	1	Kottke et al. (2016)
Batch Active Learning by Diverse Gradient Embeddings (BADGE)	✓	✗	✗	✓	1	Ash et al. (2020)
Clustering Uncertainty-weighted Embeddings (CLUE)	✓	✗	✗	✓	1	Prabhu et al. (2021)
Fast Active Learning by Contrastive Uncertainty (FALCUN)	✓	✗	✗	✓	1	Gilhuber et al. (2024)
Dropout Query (DropQuery)	✓	✗	✗	✓	1	Gupte et al. (2024)

 doi:10.20944/preprints202507.0252.v1

Table A2. Stream-based query strategies implemented by `scikit-activeml`.

Name	Classi-fication	Regres-sion	Multi-annotator	Diverse Batches	Variants	References
Baselines and Wrappers						
Periodic Sampling	✓	✓	✗	✗	1	N/A
Stream Random Sampling	✓	✓	✗	✗	2	N/A
Informativeness						
Fixed-Uncertainty	✓	✗	✗	✗	1	Žliobaitė et al. (2014)
Randomized-Variable-Uncertainty	✓	✗	✗	✗	1	Žliobaitė et al. (2014)
Variable-Uncertainty	✓	✗	✗	✗	1	Žliobaitė et al. (2014)
Hybrid						
Density-Based Active Learning for Data Streams (DBALStream)	✓	✗	✗	✗	1	Ienco et al. (2014)
Split	✓	✗	✗	✗	1	Žliobaitė et al. (2014)
Probabilistic Active Learning (PAL) in Datastreams	✓	✗	✗	✗	1	Kottke et al. (2015)
Cognitive Dual-Query Strategy (CoqDQS)	✓	✗	✗	✗	5	Liu et al. (2023)

Appendix C. Comparison of Active Learning Projects

Table 1 is based on a qualitative and a quantitative review of each **active learning project's query strategies**. Qualitatively, we mark pool, stream, multi-annotator, regression, and classification as "supported" whenever a built-in strategy or example code covers the scenario. Quantitatively, we count every class or function that implements a query strategy and, when one implementation offers algorithmically distinct variants, e.g., alternative uncertainty measures in the case of uncertainty sampling, loss functions in the case of expected error reduction, or disagreement measures in the case of query by committee, we count each variant once. Mere numeric hyperparameter tweaks are ignored. Counting publications instead of code would miss baselines and wrappers, such as random sampling and articles that bundle several strategies, so the code-level tally gives the most meaningful comparison.

Table 2 refers to each **active learning project's code and documentation**. We report whether unit tests exist and, if available, their coverage percentage. We verify the presence of a tutorial, i.e., a runnable, documented example such as a Jupyter notebook that solves a concrete task, and a developer or contribution guide that provides step-by-step instructions for extending the respective active learning project. We then list external machine learning libraries, i.e., `scikit-learn` (Pedregosa et al. 2011), `pytorch` (Paszke et al. 2019), `skorch` (Tietz et al. 2017), `pytorch-lightning` (Falcon and Team 2023), `transformers` (Wolf et al. 2020), and `river` (Montiel et al. 2021), that integrate with the active learning project, counting a library as supported when tutorial code or other documentation lets users employ its models within the active learning workflow similar to built-in models.

References

Settles, B. Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Li, D.; Wang, Z.; Chen, Y.; Jiang, R.; Ding, W.; Okumura, M. A Survey on Deep Active Learning: Recent Advances and New Frontiers. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, pp. 5879–5899.

Cacciarelli, D.; Kulahci, M. Active learning for data streams: a survey. *Mach. Learn.* **2024**, *113*, 185–239.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.

Herde, M.; Huseljic, D.; Sick, B.; Calma, A. A survey on cost types, interaction schemes, and annotator performance models in selection algorithms for active learning in classification. *IEEE Access* **2021**, *9*, 166970–166989.

Ash, J.T.; Zhang, C.; Krishnamurthy, A.; Langford, J.; Agarwal, A. Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. In Proceedings of the Int. Conf. Learn. Represent., 2020.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proceedings of the Adv. Neural Inf. Process. Syst., 2019, pp. 8026–8037.

 doi:10.20944/preprints202507.0252.v1

Tietz, M.; Fan, T.J.; Nouri, D.; Bossan, B.; skorch Developers. *skorch: A scikit-learn compatible neural network library that wraps PyTorch*, 2017.

Oquab, M.; Darcet, T.; Moutakanni, T.; Vo, H.V.; Szafraniec, M.; Khalidov, V.; Fernandez, P.; Haziza, D.; Massa, F.; El-Nouby, A.; et al. DINOv2: Learning Robust Visual Features without Supervision. *Trans. Mach. Learn. Res.* **2023**.

Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the Conf. North Am. Chapter Assoc. Comput. Linguist.: Hum. Lang. Technol., 2019, pp. 4171–4186.

Baevski, A.; Zhou, Y.; Mohamed, A.; Auli, M. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. In Proceedings of the Adv. Neural Inf. Process. Syst., 2020, pp. 12449–12460.

Hacohen, G.; Dekel, A.; Weinshall, D. Active Learning on a Budget: Opposite Strategies Suit High and Low Budgets. In Proceedings of the Int. Conf. Mach. Learn., 2022, pp. 8175–8195.

Yehuda, O.; Dekel, A.; Hacohen, G.; Weinshall, D. Active Learning Through a Covering Lens. In Proceedings of the Adv. Neural Inf. Process. Syst., 2022, pp. 22354–22367.

Huseljic, D.; Hahn, P.; Herde, M.; Rauch, L.; Sick, B. Fast Fishing: Approximating Bait for Efficient and Scalable Deep Active Image Classification. In Proceedings of the Jt. Eur. Conf. Mach. Learn. Knowl. Discov. Databases., 2024, pp. 280–296.

Gupte, S.R.; Aklilu, J.; Nirschl, J.J.; Yeung-Levy, S. Revisiting Active Learning in the Era of Vision Foundation Models. *Trans. Mach. Learn. Res.* **2024**.

Huang, K.H. DeepAL: Deep Active Learning in Python. *arXiv:2111.15258* **2021**.

Yang, Y.Y.; Lee, S.C.; Chung, Y.A.; Wu, T.E.; Chen, S.A.; Lin, H.T. libact: Pool-based Active Learning in Python. *arXiv:1710.00379* **2017**.

Tang, Y.P.; Li, G.X.; Huang, S.J. ALiPy: Active Learning in Python. *arXiv:1901.03802* **2019**.

Danka, T.; Horvath, P. modAL: A modular active learning framework for Python. *arXiv:1805.00979* **2018**.

Tsvigun, A.; Sanochkin, L.; Larionov, D.; Kuzmin, G.; Vazhentsev, A.; Lazichny, I.; Khromov, N.; Kireev, D.; Rubashevskii, A.; Shahmatova, O.; et al. ALToolbox: A Set of Tools for Active Learning Annotation of Natural Language Texts. In Proceedings of the Conf. Empir. Methods Nat. Lang. Process.: Syst. Demonstr., 2022, pp. 406–434.

Atighehchian, P.; Branchaud-Charron, F.; Lacoste, A. Bayesian active learning for production, a systematic study and a reusable library. *arXiv:2006.09916* **2020**.

Scherer, P.; Pouplin, A.; Del Vecchio, A.; Bolton, O.; Soman, J.; Taylor-King, J.P.; Edwards, L.; Gaudelet, T.; et al. PyRelationAL: a python library for active learning research and development. *arXiv:2205.11117* **2022**.

Schröder, C.; Müller, L.; Niekler, A.; Potthast, M. Small-Text: Active Learning for Text Classification in Python. In Proceedings of the Conf. Eur. Chapter Assoc. Comput. Linguist.: Syst. Demonstr., 2023, pp. 84–95.

Wu, J.; Sheng, V.S.; Zhang, J.; Li, H.; Dadakova, T.; Swisher, C.L.; Cui, Z.; Zhao, P. Multi-Label Active Learning Algorithms for Image Classification: Overview and Future Promise. *ACM Comput. Surv.* **2020**, *53*, 1–35.

Li, C.Y.; Rakitsch, B.; Zimmer, C. Safe Active Learning for Multi-Output Gaussian Processes. In Proceedings of the Int. Conf. Artif. Intell. Stat., 2022, pp. 4512–4551.

Vlachos, A. A stopping criterion for active learning. *Comput. Speech Lang.* **2008**, *22*, 295–312.

Kumar, P.; Gupta, A. Active learning query strategies for classification, regression, and clustering: A survey. *J. Comput. Sci. Technol.* **2020**, *35*, 913–945.

Wu, D.; Lin, C.T.; Huang, J. Active Learning for Regression using Greedy Sampling. *Inf. Sci.* **2019**, *474*, 90–105.

Donmez, P.; Carbonell, J.G.; Schneider, J. Efficiently Learning the Accuracy of Labeling Sources for Selective Sampling. In Proceedings of the ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., 2009, pp. 259–268.

Raykar, V.C.; Yu, S.; Zhao, L.H.; Valadez, G.H.; Florin, C.; Bogoni, L.; Moy, L. Learning from Crowds. *J. Mach. Learn. Res.* **2010**, *11*, 1297–1322.

Herde, M.; Lührs, L.; Huseljic, D.; Sick, B. Annot-Mix: Learning with Noisy Class Labels from Multiple Annotators via a Mixup Extension. In Proceedings of the Eur. Conf. Artif. Intell., 2024, pp. 2910–2918.

Žliobaitė, I.; Bifet, A.; Pfahringer, B.; Holmes, G. Active Learning With Drifting Streaming Data. *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *25*, 27–39.

Lewis, D.D.; Gale, W.A. A Sequential Algorithm for Training Text Classifiers. In Proceedings of the Annu. Int. ACM-SIGIR Conf. Res. Dev. Inf. Retr., 1994, pp. 3–12.

Scheffer, T.; Decomain, C.; Wrobel, S. Active Hidden Markov Models for Information Extraction. In Proceedings of the Int. Symp. Intell. Data Anal., 2001, pp. 309–318.

 doi:10.20944/preprints202507.0252.v1

Seung, H.S.; Opper, M.; Sompolinsky, H. Query by Committee. In Proceedings of the Annu. Workshop Comput. Learn. Theory., 1992, pp. 287–294.

Engelson, S.P.; Dagan, I. Minimizing Manual Annotation Cost in Supervised Training from Corpora. In Proceedings of the Annu. Meet. Assoc. Comput. Linguist., 1996, pp. 319–326.

McCallum, A.K.; Nigamy, K. Employing EM and Pool-Based Active Learning for Text Classification. In Proceedings of the Int. Conf. Mach. Learn., 1998, pp. 359–367.

Burbidge, R.; Rowland, J.J.; King, R.D. Active Learning for Regression Based on Query by Committee. In Proceedings of the Intell. Data Eng. Autom. Learn., 2007, pp. 209–218.

Beluch, W.H.; Genewein, T.; Nürnberger, A.; Köhler, J.M. The Power of Ensembles for Active Learning in Image Classification. In Proceedings of the IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2018, pp. 9368–9377.

Cohn, D.A.; Ghahramani, Z.; Jordan, M.I. Active Learning with Statistical Models. *J. Artif. Intell. Res.* **1996**, *4*, 129–145.

Roy, N.; McCallum, A. Toward Optimal Active Learning through Monte Carlo Estimation of Error Reduction. In Proceedings of the Int. Conf. Mach. Learn., 2001, pp. 441–448.

Margineantu, D.D. Active Cost-Sensitive Learning. In Proceedings of the Int. Jt. Conf. Artif. Intell., 2005, p. 1622–1623.

Kapoor, A.; Horvitz, E.; Basu, S. Selective Supervision: Guiding Supervised Learning with Decision-Theoretic Active Learning. In Proceedings of the Int. Jt. Conf. Artif. Intell., 2007, pp. 877–882.

Joshi, A.J.; Porikli, F.; Papanikolopoulos, N. Multi-class Active Learning for Image Classification. In Proceedings of the IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2009, p. 2372–2379.

Houlsby, N.; Huszár, F.; Ghahramani, Z.; Lengyel, M. Bayesian Active Learning for Classification and Preference Learning. *arXiv:1112.5745* **2011**.

Cai, W.; Zhang, Y.; Zhou, J. Maximizing Expected Model Change for Active Learning in Regression. In Proceedings of the IEEE Int. Conf. Data Min., 2013, pp. 51–60.

Huang, K.H.; Lin, H.T. A Novel Uncertainty Sampling Algorithm for Cost-Sensitive Multiclass Active Learning. In Proceedings of the IEEE Int. Conf. Data Min., 2016, pp. 925–930.

Wang, H.; Chang, X.; Shi, L.; Yang, Y.; Shen, Y.D. Uncertainty Sampling for Action Recognition via Maximizing Expected Average Precision. In Proceedings of the Int. Jt. Conf. Artif. Intell., 2018, pp. 964–970.

Käding, C.; Rodner, E.; Freytag, A.; Mothes, O.; Barz, B.; Denzler, J. Active Learning for Regression Tasks with Expected Model Output Changes. In Proceedings of the Br. Mach. Vis. Conf., 2018.

Chen, P.L.; Lin, H.T. Active Learning for Multiclass Cost-Sensitive Classification Using Probabilistic Models. In Proceedings of the Conf. Technol. Appl. Artif. Intell., 2013, pp. 13–18.

Kirsch, A.; Van Amersfoort, J.; Gal, Y. BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. In Proceedings of the Adv. Neural Inf. Process. Syst., 2019, pp. 7026–7037.

Nguyen, V.L.; Destercke, S.; Hüllermeier, E. Epistemic Uncertainty Sampling. In Proceedings of the Int. Conf. Discov. Sci., 2019, pp. 72–86.

Elreedy, D.; F. Atiya, A.; I. Shaheen, S. A Novel Active Learning Regression Framework for Balancing the Exploration-Exploitation Trade-Off. *Entropy* **2019**, *21*, 651.

Margatina, K.; Vernikos, G.; Barrault, L.; Aletras, N. Active Learning by Acquiring Contrastive Examples. In Proceedings of the Conf. Empir. Methods Nat. Lang. Process., 2021, pp. 650–663.

Sener, O.; Savarese, S. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In Proceedings of the Int. Conf. Learn. Represent., 2018.

Gissin, D.; Shalev-Shwartz, S. Discriminative Active Learning. *arXiv:1907.06347* **2019**.

Jose, A.; de Mendonça, J.P.A.; Devijver, E.; Jakse, N.; Monbet, V.; Poloni, R. Regression Tree-based Active Learning. *Data Min. Knowl. Discov.* **2023**, pp. 420–460.

Donmez, P.; Carbonell, J.G.; Bennett, P.N. Dual Strategy Active Learning. In Proceedings of the Eur. Conf. Mach. Learn., 2007, pp. 116–127.

Huang, S.J.; Jin, R.; Zhou, Z.H. Active Learning by Querying Informative and Representative Examples. In Proceedings of the Adv. Neural Inf. Process. Syst., 2010.

Huang, S.J.; Jin, R.; Zhou, Z.H. Active Learning by Querying Informative and Representative Examples. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 1936–1949.

Reitmaier, T.; Sick, B. Let us know your decision: Pool-based active training of a generative classifier with the selection strategy 4DS. *Inf. Sci.* **2013**, *230*, 106–131.

Kottke, D.; Krempl, G.; Lang, D.; Teschner, J.; Spiliopoulou, M. Multi-class Probabilistic Active Learning. In Proceedings of the Eur. Conf. Artif. Intell., 2016, pp. 586–594.

 doi:10.20944/preprints202507.0252.v1

Prabhu, V.; Chandrasekaran, A.; Saenko, K.; Hoffman, J. Active Domain Adaptation via Clustering Uncertainty-weighted Embeddings. In Proceedings of the IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 8505–8514.

Gilhuber, S.; Beer, A.; Ma, Y.; Seidl, T. FALCUN: A Simple and Efficient Deep Active Learning Strategy. In Proceedings of the Jt. Eur. Conf. Mach. Learn. Knowl. Discov. Databases, 2024, pp. 421–439.

Ienco, D.; Žliobaitė, I.; Pfahringer, B. High density-focused uncertainty sampling for active learning over evolving stream data. In Proceedings of the Int. Workshop Big Data Streams Heterog. Source Min. Algorithms Syst. Program. Models Appl., 2014, pp. 133–148.

Kottke, D.; Krempl, G.; Spiliopoulou, M. Probabilistic Active Learning in Datastreams. In Proceedings of the Adv. Intell. Data Anal., 2015, pp. 145–157.

Liu, S.; Xue, S.; Wu, J.; Zhou, C.; Yang, J.; Li, Z.; Cao, J. Online Active Learning for Drifting Data Streams. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *34*, 186–200.

Falcon, W.; Team, T.P.L. PyTorch Lightning, 2023. Accessed 06-05-2024, https://doi.org/10.5281/zenodo.8250019.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. Transformers: State-of-the-Art Natural Language Processing. In Proceedings of the Conf. Empir. Methods Nat. Lang. Process.: Syst. Demonstr., 2020, pp. 38–45.

Montiel, J.; Halford, M.; Mastelini, S.M.; Bolmier, G.; Sourty, R.; Vaysse, R.; Zouitine, A.; Gomes, H.M.; Read, J.; Abdessalem, T.; et al. River: machine learning for streaming data in Python. *J. Mach. Learn. Res.* **2021**, *22*, 1–8.