stats

MDPI

Article

Active Learning for Stacking and AdaBoost-Related Models

Qun Sui * and Sujit K. Ghosh

Department of Statistics, North Carolina State University, 2311 Stinson Dr, Raleigh, NC 27695-8203, USA; sujit.ghosh@ncsu.edu
* Correspondence: qsui@ncsu.edu

Abstract: Ensemble learning (EL) has become an essential technique in machine learning that can significantly enhance the predictive performance of basic models, but it also comes with an increased cost of computation. The primary goal of the proposed approach is to present a general integrative framework that allows for applying active learning (AL) which makes use of only limited budget by selecting optimal instances to achieve comparable predictive performance within the context of ensemble learning. The proposed framework is based on two distinct approaches: (i) AL is implemented following a full scale EL, which we call the ensemble learning on top of active learning (ELTAL), and (ii) apply the AL while using the EL, which we call the active learning during ensemble learning (ALDEL). Various algorithms for ELTAL and ALDEL are presented using Stacking and Boosting with various algorithm-specific query strategies. The proposed active learning algorithms are numerically illustrated with the Support Vector Machine (SVM) model using simulated data and two real-world applications, evaluating their accuracy when only a small number instances are selected as compared to using full data. Our findings demonstrate that: (i) the accuracy of a boosting or stacking model, using the same uncertainty sampling, is higher than that of the SVM model, highlighting the strength of EL; (ii) AL can enable the stacking model to achieve comparable accuracy to the SVM model using the full dataset, with only a small fraction of carefully selected instances, illustrating the strength of active learning.

Keywords: machine learning; ensemble learning; classification; AdaBoost

Citation: Sui, Q.; Ghosh, S.K. Active Learning for Stacking and AdaBoost-Related Models. *Stats* **2024**, *7*, 110–137. https://doi.org/10.3390/stats7010008

Academic Editor: Wei Zhu

Received: 12 December 2023
Revised: 15 January 2024
Accepted: 22 January 2024
Published: 24 January 2024

1. Introduction

Basic machine learning models may not always be capable of achieving the desired level of performance. Following are some examples: (i) Certain models may excel in capturing specific features, while others may perform better in other feature domains; (ii) When only a limited number of training instances are available, the effectiveness of traditional machine learning models can be severely constrained; (iii) When faced with complex decision boundaries or data patterns that cannot be effectively modeled by a single model, ensemble techniques and other more sophisticated machine learning approaches may be required.

An illustration of such a scenario is presented in Figure 1, where a total of 26 instances to be used in the training phase are shown in blue. Each instance is characterized by a pair of covariates and belongs to one of three different classes, represented by distinctive shapes such as circles, squares, and triangles. To explore the impact of training set selection on the performance of machine learning models, three separate training sets were generated by randomly drawing 12 instances from the pool of 26 for use training a model. The selected instances in each subfigure were identified and color-coded as magenta, yellow, and green, respectively. The trained models utilized these instances to develop individual hypotheses, each of which is depicted using the corresponding color. While these hypotheses are generally effective in predicting the classes of the instances, none provides a complete and accurate characterization of the underlying decision boundary. On the other hand,

by combining all three hypotheses using an averaging approach, the resulting ensembled hypothesis is able to reduce the variances inherited from the basic learners effectively.

In the machine learning field, classification models of the same or different types could be put together to build a more powerful model. This approach is referred to as ensemble learning. In essence, Ensemble Learning (EL) aims to improve the performance and robustness of a model by combining the predictions of multiple base models. EL is often employed when dealing with complex datasets or to enhance the generalization ability of models.

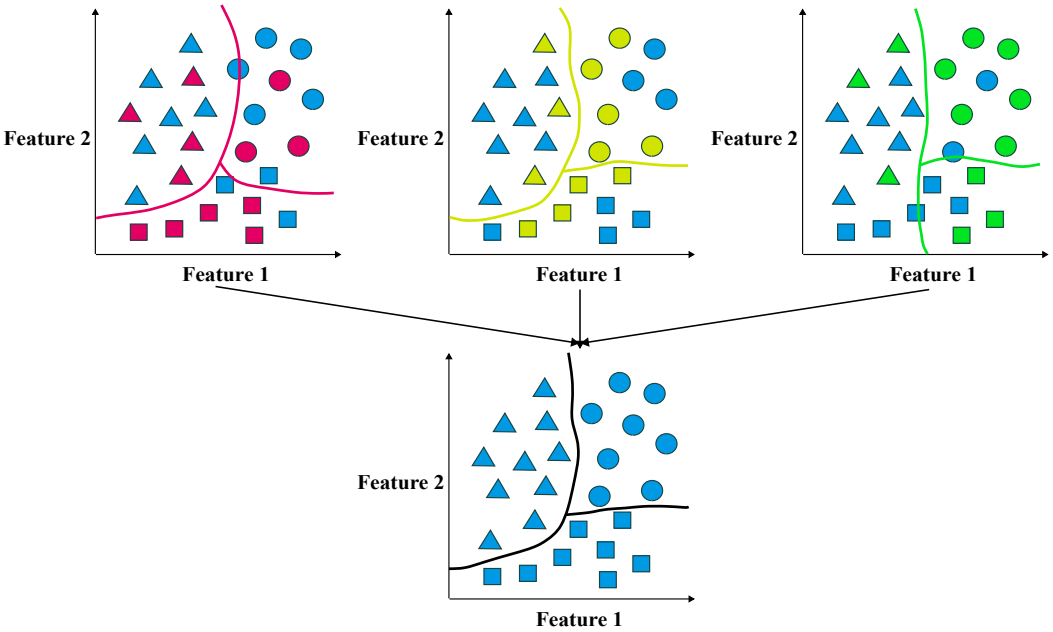

Figure 1. An illustration of ensemble learning model.

Several studies have investigated the strengths and limitations of ensemble learning. Notable benefits of ensemble learning include improved predictive accuracy, reduced variance and bias for the estimated parameters [1]. Ensemble learning can leverage the unique strengths of different model types to develop a more comprehensive understanding of the underlying data patterns. However, the results of this approach can be challenging to interpret as it may be difficult to calibrate the individual parameters from multiple models. Additionally, ensemble learning can be computationally expensive in terms of time and operational cost in practice.

Although EL is a well developed field of research, for completeness, we provide a quick glimpse of the EL methods. Traditional ensemble learning approaches fall into two categories: Sequential and Parallel. As for sequential ensemble methods, basic learners are dependent on the results from the previous ones. AdaBoost [2–5] and its boosting-related variants are some of the most popular sequential ensemble methods. In Boosting, the subsequent basic model corrects the error made by its predecessor. Weights of previously mislabeled examples are increased and weights of previously correctly labeled examples are decreased. On the other hand, there is no interdependence among the basic learners in parallel ensemble learning techniques. Multiple models can be trained concurrently and separately. To obtain the final estimation, further integration must be chosen to merge the hypotheses produced by each learner. For classification problems, if each basic learner's decision regarding an instance's class falls within the range of 0 and 1, possible integration mechanisms include the mean, weighted average, or the product of the decisions. In addition to using a function based on those basic decisions, the integration rule could also be learned via an additional machine learning model. Since different machine learning models are stacked on different levels, this technique is referred to as stacking [6,7].

Regardless of the specific updates applied to subsequent learners in Boosting or the particular integration techniques chosen in parallel approaches, ensemble learning models inevitably incur additional computational burden. Moreover, in situations where there are constraints on the resources available for labeling, but where the benefits of ensemble learning models are still desirable, the active learning framework can offer a viable solution. The primary focus of Active Learning (AL) is on reducing the amount of labeled data needed for training by intelligently selecting the most informative instances for labeling. AL is beneficial when labeling data is expensive or time-consuming. It is often used in scenarios where there is a large pool of unlabeled data, and the model is allowed to query the most valuable instances for labeling.

The majority of active learning approaches, including random sampling, uncertainty sampling [8,9], and expected error reduction (EER) [10], adopt a model-free approach in which instances are selected based on predicted probabilities or data structure. As a consequence, active learning algorithms are not limited to simple logistic regression models and can be applied to a diverse range of more complex models, including deep neural networks [11], classification trees, and gradient boosting machines [5].

Recent years have witnessed a growing interest in leveraging Active Learning (AL) in the context of deep learning models, particularly owing to the remarkable performance of these models on complicated tasks [12,13]. A variety of techniques have been proposed that combine deep learning and active learning, including Deep Bayesian Active Learning (DBAL) [14], uncertainty estimates using deep ensembles [15], geometry-based approaches [16], and Deep Ensemble Bayesian Active Learning (DEBAL) [17].

Despite extensive research for EL and AL for basic learners and convolutional neural networks, the topic of integrating AL within the EL has largely remained unexplored. Specifically, this study focuses on active learning techniques in the context of Boosting and Stacking. While ensemble learning aims to combine multiple hypotheses using a fixed dataset, active learning involves sequentially selecting instances from the pool. A crucial question that arises pertains to the timing of instance selection. When the training data is updated during ensemble learning, it is not clear whether the learners can still be improved. In our proposed work, this scenario will be referred to as active learning during ensemble learning (ALDEL). Conversely, if instance selection is performed after the integration of basic learners, the efficiency of active learning may be limited. This scenario will be referred to as ensemble learning on top of active learning (ELTAL).

Figure 2 highlights the mechanism of shifting from active learning in basic models to active ensemble learning, with a particular emphasis on classification problems. The red parts in Figure 2 denote the key aspects of this study.

To the best of our knowledge, we provide the first general framework for integrating active learning within ensemble learning. Our exposition is specifically for AdaBoost-related and Stacking-related models. We consider the ELTAL and ALDEL cases separately for both models and propose four different general algorithms, one for each setting in detail. For ELTAL algorithms, we apply active learning query strategies directly to the ensemble learner. For ALDEL in Boosting, we select instances that maximize the exponential loss and present them to the oracle for labeling. After labeling, we assign a pseudo weight to these instances for use in subsequent iterations. For ALDEL in Stacking, we divide query strategies into two categories: two-level selection and direct selection. In two-level selection, each lower-level model independently selects a candidate based on its own criterion. A second-level algorithm then chooses the final instance from these candidates. In direct selection, we treat the lower-level model's output as a transformation from the input space to a range between 0 and 1. This probabilistic output, which is more informative than raw input data, is then subjected to traditional active learning algorithms.

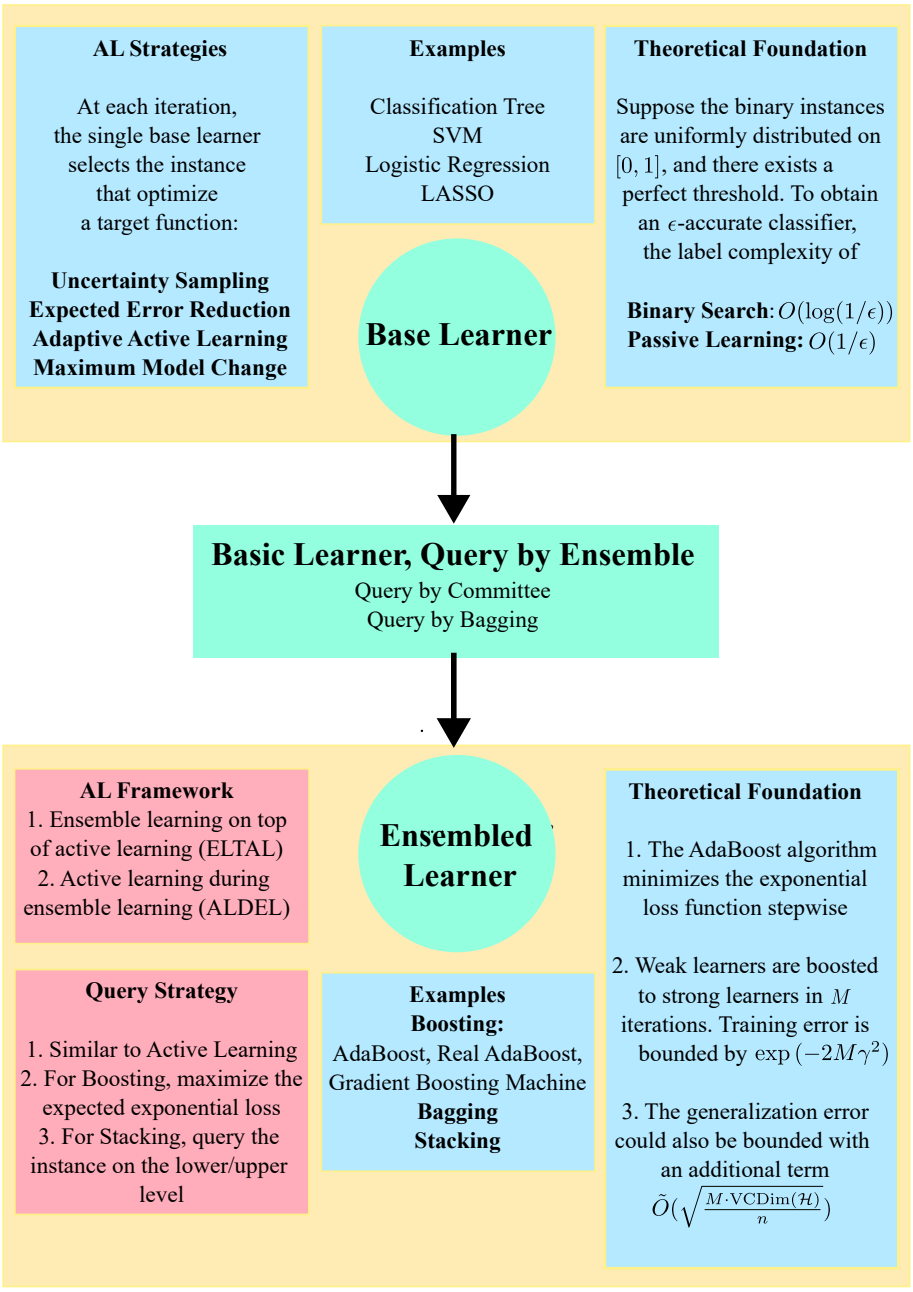

Figure 2. A transition from classic active learning to ensemble active learning.

The structure of this article is outlined as below: In Section 2, we present a brief review of the sequential and parallel ensemble learning approaches. In Section 3, we provide a general overview of active learning, the challenges and key points of combining active learning with ensemble learning. In Sections 4 and 5, we introduce our proposed active learning technique in AdaBoost-related and stacking models, respectively. In Section 6, we numerically illustrate the performance of different ensemble active learning algorithms using simulated datasets and real-world applications. In Section 7, we conclude this article by discussing the strength and weaknesses of our work.

2. Ensemble Learning: A Brief Review

Suppose a set of possibly vector valued observations $x_1, x_2, \ldots, x_n$ arise independently from a distribution $D_{\mathcal{X}}$. Let $y_1, y_2, \ldots, y_n \in \{-1, +1\}$ denote the corresponding labels. We assume that the pairs $(x_i, y_i), i = 1, 2, \ldots, n$ arise independently from a joint distribution $D_{\mathcal{X}\mathcal{Y}}$. Denote the collection of different active learning criteria such as uncertainty

sampling [8], expected error reduction [10] as $\mathcal{A}$. The hypotheses come from the set of mappings, $\mathcal{H} : D_{\mathcal{X}} \to \{-1, +1\}$. We use the function err to denote the training error of a classifier h based on full sample $S_n = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, i.e.,

$$\text{err}_{S_n}(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{h(x_i) \neq y_i\}$$

Denote the ground truth classifier as $h_0 \in \mathcal{H}$, such that $h_0(x) = y, \forall x, y \in D_{\mathcal{X}\mathcal{Y}}$. In other words, $\text{err}_{S_n}(h_0) = 0$.

Following the same setting as typically used for Probably Approximately Correct (PAC) learning [18], we define the concepts of weak learning and strong learning. A strong learner could attain arbitrarily low error as long as enough instances are provided. For any given $\epsilon > 0$ and $\delta < \frac{1}{2}$, a strong learner would return a classifier h such that

$$\Pr\{\text{err}_{S_n}(h) < \frac{1}{2} - \epsilon\} > 1 - \delta \tag{1}$$

with n =Poly($\frac{1}{\epsilon}$, $\frac{1}{\delta}$, VCDim($\mathcal{H}$)) instances, where VCDim($\mathcal{H}$) denotes the VC dimension of $\mathcal{H}$. A weak learning algorithm output a classifier h that performs better than random guessing. In other words, there exists $\epsilon_0 > 0$ and $\delta_0 < \frac{1}{2}$, such that (1) holds.

Generally speaking, the weakness of a learning algorithm is associated with the magnitude of bias or variance. The core of ensemble learning is to convert a weak learner to a strong learner by reducing either the biases or the variances of the weak learner.

2.1. Boosting

The core idea of boosting-type methods is combining different learners, where the next learner corrects the previous learner's error. The first boosting algorithm was proposed by [19]. It provides three classifiers, where the second classifier is trained with a sample that includes at least half of the misclassified samples by the first learner. The third learner is trained on the instances in which the first two learners disagree. The most famous boosting algorithm is AdaBoost [2–5], which is presented in Algorithm 1. Here, we use M as the number of basic classifiers used for boosting. In other words, the algorithm is halted after M iterations. The choice of M depends on the early stopping of a boosting algorithm. As larger M may lead to larger variances, numerous studies, such as [20], suggest that selecting a large M may result in overfitting. Therefore, M should be chosen to ensure the algorithm converges, as noted in [21]. Practically, this parameter is often determined by data-derived early stopping rules and empirical testing. In Algorithm 1, M is treated as an input parameter, whereas in parallel ensembling techniques that are introduced in the following Sections, M is automatically determined by the number of parallel basic learners selected by the user.

Algorithm 1 AdaBoost Algorithm

input : $S_n = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$: a realization of $D_{\mathcal{X}\mathcal{Y}}$; M : pre-determined number of boosted classifiers

output: Final classifier $H \in \mathcal{H}$

The initial weights are assigned as $w_{i,1} \leftarrow \frac{1}{n}$ for $i = 1, 2, \ldots, n$.

for $m \leftarrow 1$ **to** M **do**

- $h_m \leftarrow$ TrainModel(S_n) with weights $W_m = \{w_{1,m}, w_{2,m} \ldots, w_{n,m}\}$
- $e_m \leftarrow \sum_{i=1}^{n} w_{i,m} \mathbb{1}\{h_m(x_i) \neq y_i\}$
- $\alpha_m \leftarrow \frac{1}{2} \log \frac{1-e_m}{e_m}$
- $w_{i,m+1} \leftarrow w_{i,m} \exp(-\alpha_m y_i h_m(x_i))/v_m$,where v_m is a normalizing factor that is the sum of the weights.

end for

$H \leftarrow sign(\sum_{m=1}^{M} \alpha_m h_m)$

In order to keep the weight α_m as positive, the error for the classifier at each iteration can not exceed 0.5. This matches the assumption that the learner to be boosted by the algorithm should be a weak learner.

Theorem 1 ([4], Theorem 6)**.** *If the m-th hypothesis is better than a random guessing by γ_m, i.e., $err_{S_n}(h_m) = \frac{1}{2} - \gamma_m$, then*

$$err_{S_n}(H) \leq \exp(-2\sum_{m=1}^{M}\gamma_m^2) \tag{2}$$

If γ_m's are bounded below by γ, the upper bound of (2) could be replaced by $\exp(-2M\gamma^2)$. Another theorem has shown that the upper bound could be replaced by $2^M\prod_{m=1}^{M}\sqrt{(\frac{1}{2}-\gamma_m)(\frac{1}{2}+\gamma_m)}$. With the assumption that all γ_m's are less than $\frac{1}{2}$. the AdaBoost procedure has enabled the outputted hypothesis to attain arbitrarily low training error with sufficient instances. Other than training error, the generalization error of the outputted hypothesis could also be bounded from above.

Theorem 2 ([4], Theorem 7)**.** *With high probability, the generalization error of the outputted hypothesis could be bounded as*

$$err_{D_{\mathcal{X}\mathcal{Y}}}(H) \leq err_{S_n}(H) + \tilde{O}(\sqrt{\frac{M \cdot VCDim(\mathcal{H})}{n}}) \tag{3}$$

Following the same idea of boosting, some variants are proposed. In Discrete Adaboost, the output of each hypothesis is binary as either -1 or 1. Schapire and Singer [22] introduced a generalized version of AdaBoost such that the output of the hypothesis could be real-valued instead of being restricted in $[-1, 1]$. Friedman et al. [23] viewed the AdaBoost-related procedure as fitting an additive model to a target function. Denote the exponential loss function as

$$J(H) = E[e^{-yH(x)}],\ y \in \{-1, +1\}. \tag{4}$$

Specifically, Friedman et al. [23] demonstrated that Boosting can be viewed as minimizing the Equation (4) by constructing an additive model aimed at $\frac{1}{2}\log\frac{\Pr(y=1|x)}{\Pr(y=-1|x)}$, which is the minimizer of $J(H)$. With an existing ensemble hypothesis H, the update process at each iteration involves minimizing $J(H + \alpha h)$ in the subsequent iteration, w.r.t α and h. If the output of each hypothesis could be real-valued confidence, Equation (4) could be minimized by directly setting the derivative $\partial J/\partial h$ as 0. This becomes the update rule from the other algorithm Real AdaBoost proposed by Friedman et al. [23]. In Discrete AdaBoost, however, the optimization step are restricted under the condition that the output of h is binary. Friedman et al. [23] further demonstrated that the expected value of the negative log-likelihood function, after transforming the responses of the negative class instances from -1 to 0, shares the same minimizer as $J(H)$. This insight led to the development of the LogitBoost and Gentle AdaBoost algorithms. Both aim to enhance the current ensemble hypothesis by employing a Newton step to minimize the expected negative log-likelihood and $J(H)$, respectively, at each iteration.

2.2. Parallel Ensemble Learning

Suppose we have a set of hypotheses $h_1, h_2, \ldots, h_M \in \mathcal{H}$. Each of them is outputted by a single basic learner on S_n. In parallel ensemble learning, hypotheses could come from different types of models. Given an instance x, we denote that the support that it belongs to the positive class from the m-th hypothesis as $h_m(x) \in [0, 1], m = 1, 2, \ldots, M$, then the predicted label for x is taken as the maximizer of the ensembled decision $d(x)$. The generalized mean rule of a decision function is given as

$$d(x|h_1, h_2, \ldots, h_M) = \left(\frac{1}{M} \sum_{m=1}^{M} w_m (h_m(x))^{\alpha} \right)^{1/\alpha}, \quad \sum_{m=1}^{M} w_m = 1.$$

Some special cases from this rule include: (i) $\alpha = 1$ and $w_m = 1/m, \forall m$ leads to the mean rule; (ii) $\alpha = 1$ and w_m's not identical leads to the weighted mean rule; (iii) $\alpha \to 0$ and $w_m = 1/m, \forall m$ leads to the product rule; (iv) $w_m = 1/m, \forall m$ and $\alpha \to \infty$ leads to the maximum rule [24].

An alternative to using a single function to aggregate results from multiple classifiers is to construct a new machine learning model that learns the optimal combination rule. This approach allows for a more systematic and data-oriented fusion of information from multiple models. Wolpert [6] proposed a stacked generalization framework, which combined cross-validation with the training of multiple classifiers. More specifically, suppose we use a K-fold cross-validation. S_n is partitioned into K disjoint subsets $S_n^{(1)}$, $S_n^{(2)}$, ..., $S_n^{(K)}$ with roughly equal sizes. Denote $\bar{S}_n^{(k)} = S_n \backslash S_n^{(k)}$. During the training phase, each machine learning model $h_m^{(k)}$ is trained on $\bar{S}_n^{(k)}, k = 1, 2, \ldots, K$ and predicted on $S_n^{(k)}$. Denote $z_{im} = h_m^{(k)}(x_i)$, $m = 1, 2, \ldots, M$ if $x_i \in S_n^{(k)}$ and let $z_i = (z_{i1}, z_{i2}, \ldots, z_{iM})$. As predictions from M machine learning models, z_i's are treated as intermediate covariates to learn the combination rule and we call them level-1 data. Denote $CV_n^{(k)} = \{(z_i, y_i) | x_i \in S_n^{(k)}\}$ as the collection of level-1 data along with their labels for the k-th cross-validation set. Another machine learning model H is further trained on $CV_n = \cup_{k=1}^{K} CV_n^{(k)}$. Here the basic models $h_1, h_2, \ldots, h_M$ are also referred to as lower-level learners, while H is regarded as the upper-level learner. If all models are trained on the same original dataset S_n, they are likely to be highly correlated. The cross-validation procedure effectively prevents overfitting.

We use Figure 3 to visualize a typical stack learning model. Suppose we use a three-fold cross-validation on four lower-level models M_1, M_2, M_3 and M_4. After partitioning, $S_n^{(1)}, S_n^{(2)}$ and $S_n^{(3)}$ are passed to all models on the lower-levels. The predictions on the k-th fold $CV_n^{(k)}$ are combined from $h_1^{(k)}$, $h_2^{(k)}$, $h_3^{(k)}$ and $h_4^{(k)}$. The intermediate covariates are then vertically concatenated as CV_n to feed H.

After the proposal of stacked generalization in [6], Breiman [7] discussed its eligibility in regression models and illustrated with some simulation examples. As another combining strategy, Another approach, Bayes Model Averaging (BMA), assigns weights to different learners based on their posterior probabilities, a method investigated in studies by [25–27]. Clarke [28] has demonstrated that BMA is not worse than stacking under the assumption that the correct data generating model (DGM) is on the list of models under consideration and the bias is relatively low. However, this scenario is too idealistic in practice. Generally, stacking is more robust and superior to BMA [29].

Ensemble learning, while powerful, faces challenges with increasing computational demands due to the necessity of training and integrating multiple models. The first challenge lies in the training of multiple models; this process can be particularly time-intensive for complex models or when dealing with large datasets. Additionally, combining the outputs of these models further escalates computational requirements. For instance, in AdaBoost algorithms, recalculating weights for all instances in each iteration and deriving the final prediction from a weighted average of all model predictions adds to the computational load. In parallel ensemble learning methods, the integration of support from each lower-level model through a decision function also increases computational complexity. This complexity is further amplified in stacking-based approaches, where an upper-level model must be trained to learn the combination rule, adding to the training burden. This motivates us to leverage active learning from traditional machine learning models to ensemble learning models, which can substantially reduce the labeling and computational cost during the training process.

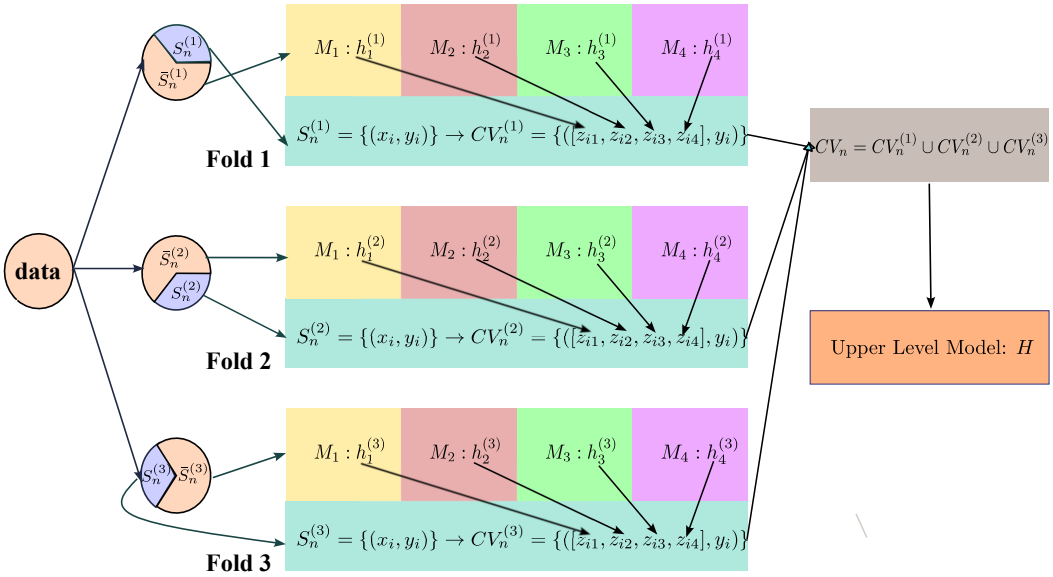

Figure 3. An illustration of cross-validated stacking model.

3. Active Learning for Ensemble Learning: A Brief Review

Active learning for basic machine learning models, specifically classification models, has been investigated in a huge variety of literature [8–10,30]. At the heart of active learning is the notion of starting the training process with a limited number of labeled instances, and subsequently selecting unlabeled instances for annotation using an active learning algorithm. Instances can be selected either one at a time or in batches. Once an instance has been selected, it is presented to an oracle for labeling, and subsequently added to the labeled dataset for retraining the model.

Under the pool-based active learning setting, all available candidate instances to be labeled come from a pool S_n. Denote L_1 as the initially labeled dataset, which is a subset of S_n and U_1 as the initially unlabeled dataset. At the t-th step, the machine learning model is trained on L_t and outputs a hypothesis $h_t \in \mathcal{H}$. Suppose we have an active learning algorithm $A \in \mathcal{A}$. A selects an instance ξ_t from U_t based on some querying strategy. In most occasions, the querying strategy can be quantified by a selection function $g(\cdot|L_t, h_t)$. After ξ_t being labeled as η_t by some human experts, both the instance and its label are added to L_t and removed from U_t. The selection and training phases are processed iteratively until the stopping criterion is satisfied. In the following Sections, we use the notation T as the total number of iterations that the active learning is processed until halted.

The primary concern for designing an active learning algorithm $A \in \mathcal{A}$ is to pick an optimal selection criterion such that the most informative instances are queried for labeling. Various criteria have been employed to classify distinct querying strategies. Kumar and Gupta [31] proposed a classification of query strategies in the context of pool-based active learning. These strategies can be broadly categorized into informative-based and representative-based approaches, with some algorithms attempting to balance the two categories through a combination of both. Informative-based strategies prioritize the selection of instances that are most informative to the current model, such as those exhibiting high levels of uncertainty [8] or resulting in maximum model change [32]. Representative-based strategies, on the other hand, consider the structure of the unlabeled dataset and the relationships between its instances when selecting queries. This was also referred to as active learning based on instance correlations in Fu et al. [33] and was further categorized into exploration on feature correlation, label correlation, feature and label correlation and graph structure. Examples of representative-based strategies include density-based approaches [34]. In order to balance informativeness and representativeness, some algorithms use tradeoff functions to quantify both properties and select instances based on their combined score. The precise criteria for selection

can vary depending on the algorithm and its specific objectives. This tradeoff function is introduced as the utility function in Fu et al. [33], which is the product of uncertainty and correlation measures. Some examples of this combination approach include Adaptive Active Learning (AAL) [35] and Maximizing Variance for Active Learning (MVAL) [36].

The task of retraining classification models becomes particularly challenging in the context of ensemble learning, where the use of multiple models can result in a significant increase in computational complexity. Active learning algorithms that rely on retraining classification models can exacerbate this problem.

In this article, we propose two directions that employ active learning in ensemble learning. The first approach, we refer to as **Active Learning during Ensemble Learning (ALDEL)**, involves applying active learning algorithms to each base learner individually. The second approach, we call **Ensemble Learning on top of Active Learning (ELTAL)**, involves applying active learning algorithms directly to the predictions from the upper learner, selecting a single instance at each iteration. Figures 4 and 5 give a visualization of these two different directions of active learning in boosting and stacking, respectively.

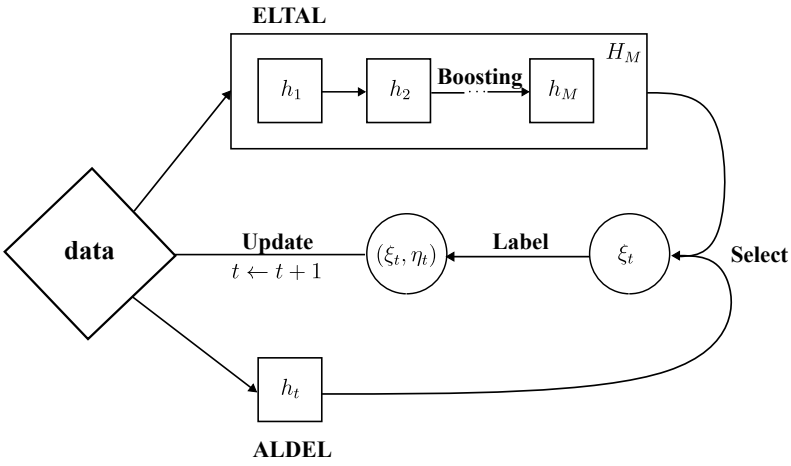

Figure 4. A visualization of ALDEL in Boosting and ELTAL in Boosting.

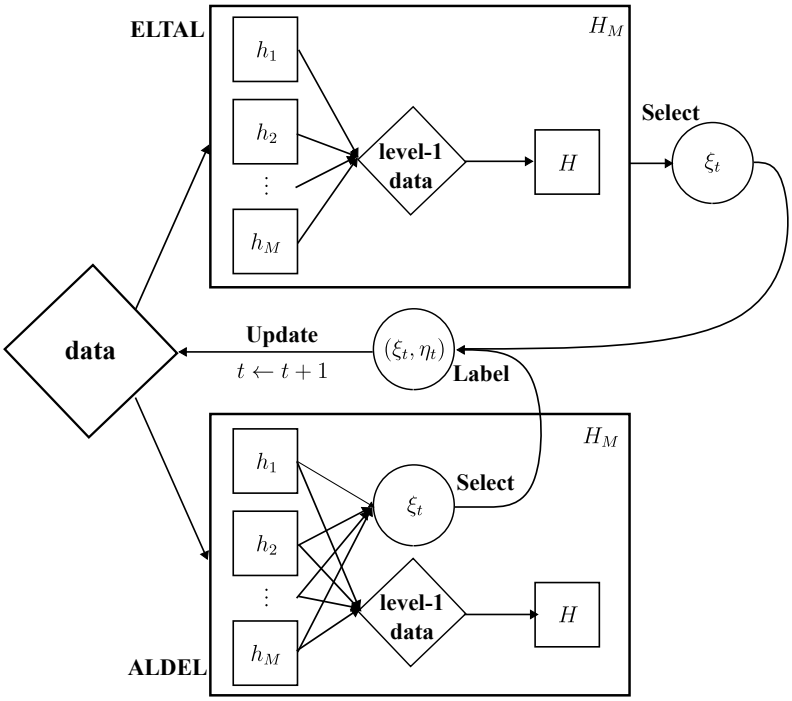

Figure 5. A visualization of ALDEL in Stacking and ELTAL in Stacking.

4. Active Learning in Boosting

4.1. ELTAL in Boosting

To the best of our current knowledge, ActiveBoost was the first Boosting Algorithm to employ an active learning framework, as described by Wang et al. [37]. The authors were inspired by the stability of the naïve Bayes classifier when applied to the boosting process. Notably, the algorithm selects instances to be labeled after the boosting procedure, and therefore can be classified as an ELTAL approach. A disadvantage of this technique is that it necessitates the creation of numerous additional models without considering the underlying structure of the data.

This study presents a systematic and comprehensive approach to address the challenges of combining active learning with boosting. Our proposed algorithm, outlined in Algorithm 2, offers a more general solution to this problem. In recent years, a wide range of boosting procedures have been proposed, including Discrete AdaBoost, Real AdaBoost, Logit Boost, and Gentle AdaBoost, which all result in binary output hypotheses $H = \text{sign}(\sum_{m=1}^{M} \alpha_m h_m)$. To obtain a more detailed understanding of each candidate's response towards H, we eliminate the $\text{sign}(\cdot)$ function, and intermediate weak learners' results are required to be continuous to align with the proposed algorithm. As an example, consider the Real AdaBoost algorithm. At each iteration, the hypothesis is derived as $h_m(x) = \frac{1}{2}\log\{p_m(x)/(1-p_m(x))\}$, where $p_m(x)$ is the probability estimate that an instance x belongs to the positive class when the learner is trained on L_t with W_m. Weights of all hypotheses are identical in Real AdaBoost, while the weights of labeled instances are updated as $w_{i,m} \leftarrow w_{i,m}\exp\{-y_i h_m(x_i)\}$ before normalization.

Algorithm 2 Ensemble Learning on top of Active Learning (ELTAL) in Boosting

input : labeled dataset L_1, unlabeled dataset U_1, active learning algorithm $A \in \mathcal{A}$ and its induced selection function g.
output: L_T, U_T, final hypothesis $H_T \in \mathcal{H}$
for $t \leftarrow 1$ **to** T **do**
  The initial weights are assigned as $W_1 = \{\frac{1}{|L_t|}, \frac{1}{|L_t|}, \ldots, \frac{1}{|L_t|}\}$.
  for $m \leftarrow 1$ **to** M **do**
    $h_m \leftarrow \text{TrainModel}(L_t)$ with weights W_m
    $\alpha_m \leftarrow \text{GetWeight}(W_m, L_t, h_m)$
    $W_{m+1} \leftarrow \text{UpdateWeight}(W_m, L_t, h_m)$.
  end for
  $H_t \leftarrow \sum_{m=1}^{M} \alpha_m h_m$
  $\xi_t \leftarrow g(U_t|L_t, H_t)$, $\eta_t \leftarrow \text{Query}(\xi_t)$
  $L_{t+1} \leftarrow L_t \cup \{(\xi_t, \eta_t)\}$
  $U_{t+1} \leftarrow U_t \backslash \{\xi_t\}$
end for

ELTAL algorithms select unlabeled instances when the ensembling procedure is complete. Although the information used for active learning is obtained from a strong learner, the computational burden is enormous since the ensembling procedure needs to be repeated after each update of the training set. ELTAL framework trains the model before implementing active learning. This non-overlapping structure of the framework is similar to the active learning approach used for basic machine learning models, thus provides a smooth transition from active learning for basic learners to ensembled learners. When the output of ensembled hypothesis is probabilistic, query strategies can be leveraged to identify the most informative samples. In this work, popular active learning approaches such as uncertainty sampling and expected error reduction are considered as selection criteria. To illustrate this, we provide some examples of the selection function g in ELTAL in Boosting.

- Uncertainty Sampling [8]:

$$g_1(U_t|L_t, H_t) = \arg\min_{x \in U_t} \max\{H_t(x), 1 - H_t(x)\} \tag{5}$$

$$g_2(U_t|L_t, H_t) = \arg\min_{x \in U_t} \{H_t(x) \log H_t(x) + (1 - H_t(x)) \log(1 - H_t(x))\} \tag{6}$$

- Expected Error Reduction [10]:

$$g_{\text{EER}}(U_t|L_t, H_t) = \arg\min_{x \in U_t} \Bigg\{ \sum_{u \in \{-1,+1\}} \Pr(\text{sign}(H_t(x)) = u|x, L_t) \sum_{x_i \in U_t} \sum_{v \in \{-1,+1\}} \Pr(\text{sign}(H_t(x_i)) = v|x_i, L^u_{t,x}) \log \Pr(\text{sign}(H_t(x_i)) = v|x_i, L^u_{t,x}) \Bigg\}, \tag{7}$$

where $\Pr(\text{sign}(H_t(x)) = u|x, L_t)$ is the probability that an instance x belongs to the class u when the hypothesis H_t is trained on L_t. The notation $L^u_{t,x} = L_t \cup \{(x, u)\}$ is the extended labeled dataset when the instance is labeled as u and added to L_t.

4.2. ALDEL in Boosting

For traditional AdaBoost-based algorithms, a new hypothesis is constructed at each iteration using all training data, with the weight function used to gauge the significance of each instance. If the i-th instance's weight $w_{i,m}$ is close to zero, it has minimal impact on h_m. Drawing motivation from this insight, we can initiate the boosting process by utilizing only a subset of the available training data. This subset can be treated as the labeled dataset within an active learning framework, with the remaining instances composing the unlabeled dataset. The incorporation of each instance into the subset can be determined based on whether its weight is zero. The indices of S_n are rearranged to ensure that the first $|L_m|$ instances are labeled, and their weights are equally assigned as $1/|L_m|$. This approach falls under the purview of ALDEL, which combines active learning and ensemble learning in a unified manner.

ALDEL in Boosting presents two significant challenges. Firstly, because the weight of the newly added instance has been zero until the last iteration, an approximate weight must be assigned to enable updates in subsequent iterations. Secondly, an appropriate selection criterion that aligns with the boosting objective must be chosen. Regardless of these two concerns, a general framework of ALDEL in Boosting is presented in Algorithm 3 and Figure 6 provides a visualization of Algorithm 3.

Algorithm 3 Active Learning During Ensemble Learning (ALDEL) in Boosting

input : $L_1 = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{|L_1|}, y_{|L_1|})\}, U_1 = \{(x_{|L_1|+1}, z_{|L_1|+1}), \ldots, (x_n, z_n)\}$, where y_i's are known labels and z_j's are unknown labels. An active learning algorithm $A \in \mathcal{A}$ and its induced selection function g

output: Final classifier $H_M \in \mathcal{H}$

$W_1 \leftarrow \{w_{i,1}\}_{i=1}^n$, where $w_{i,1} = \frac{1}{|L_1|}$ for $i = 1, 2, \ldots, |L_1|$. and $w_{i,1} = 0$ otherwise.

for $m \leftarrow 1$ **to** M **do**

  $h_m \leftarrow \text{TrainModel}(S_n)$ with weights W_m

  $\xi_m \leftarrow g(U_m|L_m, h_m), \eta_m \leftarrow \text{Query}(\xi_m)$

  $w_0 \leftarrow \text{GetWeight}(h_m, \xi_m, \eta_m, L_m, W_m)$

  $W_m \leftarrow W_m \cup \{w_0\}$

  Rearrange the order of instances in S_n and weight in W_m such that the newly annotated instance has an index of $|L_{m+1}|$ and all unannotated instances are pushed back.

  $W_{m+1} \leftarrow \text{UpdateWeight}(W_m, L_m, h_m)$.

end for

$H_M \leftarrow sign(\sum_{m=1}^M h_m)$

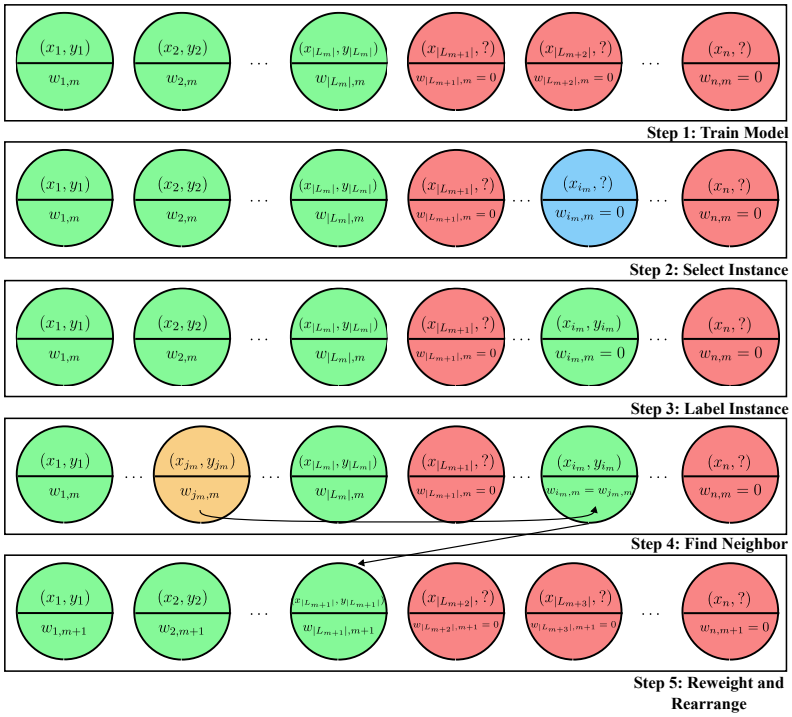

Figure 6. A visualization of ALDEL in Boosting.

The labeled instances are designated by the color green, while the unlabeled instances are colored red. In each iteration, the selected instance is colored blue, and the labeled instance utilized to match the weight is colored orange. After the completion of each iteration, the instances' weights, orders, and colors are updated.

To fulfill the definition of "neighbor", it's essential to define a metric for measuring the distance between two instances. The underlying intuition is that if two instances yield similar exponential losses relative to the current hypothesis, they are probably of comparable importance to the boosting process. This similarity allows us to use the weight of one instance as a "proxy" or "fake" weight for the newly added instance. More precisely, we determine the distance between two instances based on the difference in their contributions to the exponential loss, as calculated with respect to the current hypothesis. This approach provides a practical means to assess the similarity between instances in the context of boosting, i.e.,

$$d(x_i, x_j|h_m) = |\exp(-y_i h_m(x_i)) - \exp(-y_j h_m(x_j))| \tag{8}$$

In Real AdaBoost, it could also be rewritten as

$$d(x_i, x_j|h_m) = \left| \left(\frac{p_m(x_i)}{1 - p_m(x_i)} \right)^{\frac{-y_i}{2}} - \left(\frac{p_m(x_j)}{1 - p_m(x_j)} \right)^{\frac{-y_j}{2}} \right|$$

An instance's importance is determined by its current weight, and the learning algorithm prioritizes instances that make the greatest contributions to the loss function. This notion sheds light on the following proposed AL querying strategy.

- Expected Exponential Loss Maximization (EELM): The proposed querying strategy aims to select the instance that maximizes the expected exponential loss over all unlabeled data. Within the Real AdaBoost framework, the hypothesis $h_m(x) = 1/2 \log p_m(x)/(1 - p_m(x))$ is a transformation of the weighted probability estimates. The training and transformation steps of Real AdaBoost are identical to those of Discrete AdaBoost, with the objective of approximately optimizing $J(H)$ in a stage-wise manner. The selection criterion could be described as

$$g(U_m|L_m, h_m) = \arg\max_{x \in U_m} \{p_m(x) \exp\{-h_m(x)\} + (1 - p_m(x)) \exp\{h_m(x)\}\}. \quad (9)$$

It is important to note that the expected exponential loss in (9) can be expressed as a weighted conditional expectation E_w on W_m. For ease of presentation, we continue to denote p_m as the weighted conditional probability. The selection criterion can also be viewed from an active learning perspective. By substituting the expression of $p_m(x)$, the expected exponential loss reduces to $2\sqrt{p_m(x)(1 - p_m(x))}$, which is directly proportional to the standard deviation of $p_m(x)$. Therefore, the selected instance has the highest variance among all instances in U_m. This aligns with the principle of uncertainty sampling [8].

5. Active Learning in Stacking

5.1. ELTAL in Stacking

ELTAL in Stacking is similar to ELTAL in Boosting in a way that the active learning is implemented after the ensembling procedure. Noticed that each lower-level learner is trained K times, one on each $L_t^{(k)}$. We define $h_{t,m}^{(k)}$ as the m-th lower-level model that is trained on $L_t^{(k)}$. To get a probabilistic prediction for an instance in U_t, we use the same averaging approach. More specifically, for an instance $x_j \in U_t$,

$$z_{jm} = h_{t,m}(x_i), \quad m = 1, 2, \ldots, M, \quad (10)$$

where $h_{t,m}(\cdot) = \frac{1}{K} \sum_{k=1}^{K} h_{t,m}^{(k)}(\cdot)$. If we denote $L_t^{'}$ as the collection $\{z_j \mid x_j \in L_t\}$ and $U_t^{'}$ as the collection $\{z_j \mid x_j \in U_t\}$, the trained lower-level hypotheses create mappings from U_t to $U_t^{'}$ and from L_t to $L_t^{'}$. In other words, $L_t^{'}$ is the concatenation of the cross-validated predictions on each fold and it is used for training the upper-level learner in the t-th iteration.

In this Algorithm, we assume that the selection function g_0 takes mapped sets $L_t^{'}$ and $U_t^{'}$ as arguments instead of L_t and U_t. Furthermore, we assume that the selection function $g_0(U_t^{'}; \theta_t | L_t^{'}, H_t)$ may depend on some extra tuning parameter $\theta_t \in \Theta$. A general framework of ELTAL in Stacking is introduced in Algorithm 4.

Algorithm 4 Ensemble Learning on top of Active Learning (ELTAL) in Stacking

input : labeled dataset L_1, unlabeled dataset U_1, active learning algorithm $A \in \mathcal{A}$ and its induced selection function g_0, number of folds K

output: L_T, U_T,final hypothesis $H_T \in \mathcal{H}$

for $t \leftarrow 1$ **to** T **do**

  Partition L_t into K mutually disjoint sets $L_t^{(1)}, L_t^{(2)}, \ldots, L_t^{(K)}$ with equal sizes and let $\bar{L}_t^{(k)} = L_t \backslash L_t^{(k)}, k = 1, 2, \ldots, K$

  for $k \leftarrow 1$ **to** K **do**

    for $m \leftarrow 1$ **to** M **do**

      $h_{t,m}^{(k)} \leftarrow \text{TrainModel}(\bar{L}_t^{(k)})$

      $z_{im} \leftarrow h_{t,m}^{(k)}(x_i)$ if $x_i \in L_t^{(k)}$

    end for

    $z_i \leftarrow (z_{i1}, z_{i2}, \ldots, z_{iM})$ if $x_i \in L_t^{(k)}$

    $CV_t^{(k)} \leftarrow \{(z_i, y_i) | x_i \in L_t^{(k)}\}$

  end for

  $L_t^{'} \leftarrow \cup_{k=1}^{K} CV_t^{(k)}$, $U_t^{'} \leftarrow \text{Mapping}(U_t, h_{t,1}, h_{t,2}, \ldots, h_{t,M})$

  $H_t \leftarrow \text{TrainModel}(L_t^{'})$

  $\xi_t \leftarrow g_0(U_t^{'}; \theta_t | L_t^{'}, H_t)$, $\eta_t \leftarrow \text{Query}(\xi_t)$

  $L_{t+1} \leftarrow L_t \cup \{(\xi_t, \eta_t)\}$

  $U_{t+1} \leftarrow U_t \backslash \{\xi_t\}$

end for

If we replace L_t and U_t by $L_t^{'}$ and $U_t^{'}$, respectively, we could still use uncertainty sampling and expected error reduction as some examples for the choices of g such as (5)–(7).

5.2. Two-Level Selection for ALDEL in Stacking

After the lower-level models are trained, ALDEL in Stacking can be applied to $U_t^{'}$ directly, or different lower-level models can individually select their preferred instances. However, this approach poses several challenges in deploying ALDEL with Stacking.

- How to determine the appropriate selection criterion for each lower-level model. As the lower-level models are heterogeneous, it is reasonable to assume that their selection functions will differ.
- How should the selected M instances be combined or integrated if each learner selects only one instance per iteration.
- How to implement the active learning framework in the context of cross-validation.

Suppose the m-th lower-level model would apply the active learning algorithm $A_m \in \mathcal{A}$ on the t-th iteration. The selection function $g_m \in \mathcal{G}$ induced by A_m takes some extra tuning parameters $\theta_{t,m} \in \Theta_m$. Similar to ELTAL in Stacking, each lower-level learner is trained on the cross-validation subset $\bar{L}_t^{(k)}$ and predicted on $L_t^{(k)}$ on the t-th iteration.

The presented framework introduces a complication stemming from cross-validation. Specifically, each lower-level model is subjected to K iterations on distinct cross-validation subsets, but the selection function is intended to rely solely on a single replica. Consequently, it becomes necessary to construct a virtual hypothesis from the set of K hypotheses obtained from cross-validations. These hypotheses are homogeneous, and approximately $(k-2)|L_t|/k$ instances are trained by every two learners simultaneously, which can be expressed mathematically as follows:

$$|L_t^{(i)} \cap L_t^{(j)}| = \frac{(K-2)|L_t|}{K}, \quad \forall i, j = 1, 2, \ldots, K$$

To integrate the results from all hypotheses, we utilize the decision function $d(x)$ that was introduced in Section 2.2 for conventional parallel ensemble models. Thus, we treat the cross-validation results as parallel hypotheses that need to be ensembled, with $h_{t,m}^{(k)}$ serving as the individual support of each hypothesis that an unlabeled instance belongs to the positive class. Given any $x \in U_t$ and the list of hypotheses $h_{t,m}^{(1)}, h_{t,m}^{(2)}, \ldots, h_{t,m}^{(K)}$, we define the "virtual" hypothesis $h_{t,m}$ as

$$h_{t,m}(\cdot) = d(\cdot | h_{t,m}^{(1)}(\cdot), \ldots, h_{t,m}^{(K)}(\cdot)),$$

where $d(\cdot | \cdots)$ denotes the decision function introduced in Section 2.2. The m-th lower-level model's chosen instance is represented by $\xi_{t,m}$, which is collected alongside the other selected instances as $C_t = \{\xi_{t,1}, \xi_{t,2}, \ldots, \xi_{t,M}\}$. When the goal is to introduce a single instance to the labeled dataset, an additional integration operation G must be conducted on C_t. However, it is important to note that the labels of the instances in C_t cannot be queried until the integration process is complete. It is not reasonable to retain only a single instance-label pair after integration when multiple instances have been labeled since this violates the general principle of active learning. To address this issue, we use the probabilistic output of each hypothesis to construct the integration function, which we refer to as the second-level selection. Algorithm 5 outlines a general framework of ALDEL in Stacking.

Next, we provide some intuition on the second-level selection in Algorithm 5. We select the instance from C_t, such that it contribute most to improving the corresponding model. Notice that the selected instance will not be trained until the next iteration. At the $(t+1)$-th iteration, the m-th model is trained on $\bar{L}_{t+1}^{(k)}$, a cross-validated set of L_{t+1} and is measured on $L_{t+1}^{(k)}$. We define $l_m(h_m(x), y) : \mathbb{R} \times \{-1, +1\} \to \mathbb{R}_+ \cup \{0\}$ as the loss function

for the m-th lower-level model. A straightforward example for this loss function could be the cross-entropy loss as

$$l_m(h_m(x), y) = -[y \log h_m(x) + (1-y) \log (1 - h_m(x))]$$

We define $\phi_m(L_t)$ as the total loss when $h_{t,m}^{(k)}$ is measured on $L_t^{(k)}$, i.e.,

$$\phi_m(L_t) = \sum_{k=1}^{K} \sum_{x \in L_t^{(k)}} l_m(h_{t,m}^{(k)}(x), y)$$

After adding $\xi_{t,m}$ to the labeled dataset, the updated loss function under the worst case would be $\max\{\phi_m(L_t \cup \{(\xi_{t,m}, 1)\}), \phi_m(L_t \cup \{(\xi_{t,m}, -1)\})\}$. We select the instance from C_t such that $\phi_m(L_t)$ is increased minimally in the worst case. In other words,

$$G_1(C_t) = \arg\min_{\xi_{t,m} \in C_t} \frac{\max_{\eta_{t,m} \in \{-1,+1\}} \phi_m(L_t \cup \{(\xi_{t,m}, \eta_{t,m})\}) - \phi_m(L_t)}{\phi_m(L_t)} \tag{11}$$

Similarly, we define $l_H(H(z), y) : [0,1] \times \{-1, +1\} \to \mathbb{R}_+ \cup \{0\}$ as a loss function such as cross-entropy for the upper-level model. we denote $\phi_H(L_t)$ as the total upper-level loss when H_t is measured on $L_t^{'}$, i.e.,

$$\phi_H(L_t) = \sum_{x \in L_t} l_H(H_t(z), y)$$

The selection criterion would select the instance such that $\phi_H(L_{t+1})$ is minimized, i.e.,

$$G_2(C_t) = \arg\min_{\xi_{t,m} \in C_t} \max_{\eta_{t,m} \in \{-1,+1\}} \phi_H(L_t \cup \{(\xi_{t,m}, \eta_{t,m})\}) \tag{12}$$

Algorithm 5 Active Learning During Ensemble Learning (ALDEL) in Stacking

input : labeled dataset L_1, unlabeled dataset U_1, a collection of active learning algorithms $\{A_1, A_2, \ldots, A_M\}$ and their induced selection functions $\{g_1, g_2, \ldots, g_M\}$
output: L_T, U_T, final hypothesis $H_T \in \mathcal{H}$
for $t \leftarrow 1$ **to** T **do**
  Partition L_t into K mutually disjoint sets $L_t^{(1)}, L_t^{(2)}, \ldots, L_t^{(K)}$ with equal sizes and let $\bar{L}_t^{(k)} = L_t \backslash L_t^{(k)}, k = 1, 2, \ldots, K$
  $C_t \leftarrow \varnothing$
  for $m \leftarrow 1$ **to** M **do**
    for $k \leftarrow 1$ **to** K **do**
      $h_{t,m}^{(k)} \leftarrow \text{TrainModel}(\bar{L}_t^{(k)})$
      $z_{im} \leftarrow h_{t,m}^{(k)}(x_i)$ if $x_i \in L_t^{(k)}$
    end for
    $h_{t,m}(\cdot) \leftarrow d(\cdot | h_{t,m}^{(1)}(\cdot), \ldots, h_{t,m}^{(K)}(\cdot))$
    $\xi_{t,m} \leftarrow g_m(U_t; \theta_{t,m} | L_t, h_{t,m})$
    $C_t \leftarrow C_t \cup \{(\xi_{t,m}, \eta_{t,m})\}$
  end for
  $z_i = (z_{i1}, z_{i2}, \ldots, z_{iM})$, $L_t^{'} \leftarrow \{(z_i, y_i)\}$ if $x_i \in L_t$
  $H_t \leftarrow \text{TrainModel}(L_t^{'})$
  $\xi_t \leftarrow G(C_t)$
  $L_{t+1} \leftarrow L_t \cup \{(\xi_t, \eta_t)\}$
  $U_{t+1} \leftarrow U_t \backslash \{\xi_t\}$
end for

5.3. Direct Selection from the Lower-Level

In addition to the aforementioned two-level selection method, a viable alternative approach is to "take a step back" during the transition from basic to ensemble models in active learning. In other words, we employ the Query by Committee (QBC) technique for ALDEL method in Stacking. QBC is an active learning methodology where the selection is quantified by the degree of disagreement among a committee of pre-trained hypotheses [9,30]. In traditional QBC, the committee members are trained on a randomly selected subset of the labeled dataset, or even be combined with the bagging method [38,39] which employs resampling with replacement on L. The rationale behind this technique stemmed from the goal of inducing disagreement among the candidate hypotheses and thereby reducing the version space, which refers to the set of all hypotheses that perfectly predict all instances in the labeled dataset.

We leverage this idea to ALDEL in Stacking, where the available hypotheses come from M lower-level models. Since the committee of hypotheses in Stacking are outputted from different lower-level models trained on cross-validation datasets, resampling or subsampling procedures prior to traditional QBC are not necessary. The aim of QBC in Stacking should be querying an instance that induces maximal disagreement among all lower-level models. This motivation could alternatively be construed as emphasizing the indispensability of a upper-level model in cases where the degree of disagreement among committee members exceeds the capacity of a basic integration. When the instance is selected directly from the lower-level, the individual selection process in Algorithm 5 is discarded and replaced by a single selection $\xi_t = g_{\text{QBC}}(U_t; \theta_t | L_t, h_{t,1}, \ldots, h_{t,M})$

The question to be determined now is the choice of the disagreement measurement. Settles [9] has suggested different choices of the disagreement measurement, including vote entropy, soft vote entropy, Kullback-Leibler (KL) divergence [40] and Jensen-Shannon divergence [41]. The choice of a disagreement metric should be customized to the specifics of the case at hand. Soft vote entropy is useful for measuring uncertainty across different classes, while KL divergence is more suited for assessing the degree of disagreement among committee members. Considering the diversity of models at the lower levels in stacking, KL divergence is the preferred method for identifying instances of disagreement at these levels, which can then be escalated for higher-level evaluation, i.e.,

$$g_{\text{QBC}}(U_t; \theta_t | L_t, h_{t,1}, \ldots h_{t,M}) = \arg\max_{x \in U_t} \frac{1}{M} \sum_{m=1}^{M} \left\{ h_{t,m}(x) \log \frac{h_{t,m}(x)}{\bar{h}_t(x)} + (1 - h_{t,m}(x)) \log \frac{1 - h_{t,m}(x)}{1 - \bar{h}_t(x)} \right\}, \quad (13)$$

where $\bar{h}_t(x) = (1/M) \sum_{m=1}^{M} h_{t,m}(x)$ is the average decision from the committee.

So far, our attention has been directed towards informative-based querying strategies as classified in Kumar and Gupta [31]. These strategies are developed using information gleaned from a single instance relative to the current learning algorithm and do not take into account the structure inherent in the original data. Due to the selection functions used, the algorithms are often inclined to choose outliers, leading to extreme values in the selection function. To address this challenge, one could employ representative-based querying strategies, as classified in Kumar and Gupta [31], or in other words, exploration of feature correlation exploration techniques in Fu et al. [33].

The density-based approach is one of the most straightforward strategies that fall into this category. This approach entails the introduction of information density, which is defined as the product of an uncertainty measure and the average similarity between a candidate instance and all other unlabeled instances. It is reasonable to conceive lower-level models as a function that maps the set D_X to $[0,1]^M$. The columns in $L_t^{'}$ and $U_t^{'}$, originating from lower-level models, effectively represent L_t and U_t respectively. Utilizing these predictions directly at the upper level makes density-based active learning (AL)

on level-1 data a more practical and logical approach compared to using original data. Moreover, predictions from lower-level models often show high correlation for instances in the same fold. While cross-validation frameworks help in preventing overfitting in stacking models, exploring the feature structure of level-1 data remains a worthwhile pursuit. We propose the information density selection as follows

$$g_{\mathrm{ID}}(U_t;\beta_t|L_t,h_{t,1},\ldots,h_{t,M}) = \arg\max_{x\in U_t} \phi_A(z)\left(\frac{1}{|U_t'|}\sum_{z'\in U_t'}\mathrm{sim}(z,z')\right)^{\beta_t}, \tag{14}$$

where ϕ_A is an uncertainty measure on the upper-level such as the information entropy $-(z\log z+(1-z)\log(1-z))$ and β_t controls the importance tradeoff between the uncertainty and the density term at the t-th iteration. $\mathrm{sim}(\cdot,\cdot)$ is a similarity measure between a candidate instance and another instance in U_t'. To better capture the correlation between features in the level-1 data, we concentrate more on direction instead of magnitude when getting the similarity. Consequently, the cosine similarity

$$\mathrm{sim}(z,z') = \frac{z*z'}{\|z\|\,\|z'\|}$$

is preferred.

6. Numerical Illustrations

6.1. Simulated Datasets

We perform a simulation to illustrate the numerical performances of the algorithms that are introduced in Sections 4 and 5 in two-class classification tasks. Here the covariates come from a normal distribution, i.e., $x_{ij}\sim N(0,1), i=1,2,\ldots,n, j=1,2,\ldots,5$, while the binary response vector is generated using the following model.

$$\Pr(y_i=1)=\{1+\exp(-p_i^3+6p_i^2-9p_i+2)\}^{-1/3},\quad i=1,2,\ldots,n,$$

where $p_i=w^Tx_i$ and the true parameter is set as $w=(1,1,1,1,1,1)^T$. We generate a set of $S=100$ simulated datasets and each simulated dataset we generate has $n=5000$ instances in total with a train test split of 70:30. In other words, for each of the 100 datasets, we use 3500 instances for training and 1500 for testing. The models with their corresponding active learning selection criteria and hyperparameters are summarized in Table 1. The weak learner to be boosted is chosen as the support vector machine (SVM) [42] with a Radial Basis Function (RBF) kernel that is defined as

$$K_{\mathrm{RBF}}(x,x') = \exp(-\gamma||x-x'||^2).$$

To incorporate active learning querying strategies into support vector machines (SVM) [42], we implement Platt scaling [43] to obtain a probabilistic output. This method involves training a logistic regression model with the distance of each sample from the decision boundary as the input and the true class labels as the output. The BOOST model is obtained by applying Real AdaBoost for 50 iterations to the SVM. Two distinct querying strategies are applied to both SVM and ELTAL in Boosting: UNIF and ENTROPY in (6). For ALDEL in Boosting, the EELM criterion in (9) is applied.

Table 1. A list of all models in the Simulation studies.

Name	Description	Active Learning Selection Criterion	Hyperparameters
SVM	Support Vector Machine with Radial Basis Function (RBF) kernel	N/A	kernel parameter γ, regularization constant C
SVM-UNIF	Same as SVM	Random selection	
SVM-Entropy	Same as SVM	Entropy variant in (6)	
BOOST	Real AdaBoost Algorithm. The basic learner is taken as SVM.	N/A	Basic setup is the same as SVM; SVM is boosted for $M = 50$ times
ELTALB-UNIF	ELTAL in Boosting using Real AdaBoost The basic learner is taken as SVM.	Random selection	
ELTALB-Entropy	ELTAL in Boosting using Real AdaBoost The basic learner is taken as SVM.	Entropy variant in (6)	
ALDEL-EELM	ALDEL in Boosting using Real AdaBoost The basic learner is taken as SVM.	EELM in (9)	
STACK	A Stacking Model. Lower-level models are taken as SVM, Random Forest, Artificial Neural Networks (ANN) and Gradient Boosting Machine (GBM). The upper-level model is a logistic regression model.	N/A	SVM: same as above Random Forest: number of trees: 100, number of variables used for splitting ANN: number of units, weight decay GBM: max depth, learning rate, minimum number of observations in each node, number of trees: 100
ELTALS-UNIF	ELTAL in Stacking, Models on both levels are the same as STACK.	Random selection	
ELTALS-ENTROPY	ELTAL in Stacking, Models on both levels are the same as STACK.	Entropy variant in uncertainty sampling	
ALDELS-ENTROPY1	ALDEL in Stacking, Models on both levels are the same as STACK.	Lower-level selection is based on entropy Upper-level selection is implemented using G_1	
ALDELS-ENTROPY2	ALDEL in Stacking, Models on both levels are the same as STACK.	Lower-level selection is based on entropy Upper-level selection is implemented using G_2	
ALDELS-QBC	ALDEL in Stacking, Models on both levels are the same as STACK.	Apply QBC to lower-level models. KL divergence as the disagreement measure	
ALDELS-DENSITY	ALDEL in Stacking, Models on both levels are the same as STACK.	Apply Information Density AL on $U_t^{'}$. Information entropy is taken as ϕ_A. Cosine similarity is used Set $\beta = 1$	

The STACK model employs four lower-level models, namely the support vector machine (SVM) with Radial Basis Function (RBF) kernel, Random Forest (RF) with 100 trees [44], Artificial Neural Networks with one hidden layer [45], and the Gradient Boosting Machine (GBM) [46]. To construct the mapping from L_t to $L_t^{'}$, we apply a 10-fold cross-validation, and obtain the "virtual" hypothesis from cross-validation as the average function. The upper-level model of the STACK model is logistic regression. In the Random Forest model, we can obtain a probabilistic output by using probability averaging. This method involves taking the average of the class probabilities predicted by each decision tree in the forest for a given input sample. Similar to ELTAL in Boosting, we apply UNIF and ENTROPY querying strategies to ELTAL in Stacking. As for ALDEL, we perform four active learning algorithms. For two-level selection, we employ Entropy in (6) as the first-level querying strategy, and G_1 in (11) and G_2 in (12) as the second-level selection criteria. Additionally, we employ Query by Committee approach in (13) to the lower-level models, utilizing KL divergence as the disagreement measure and a density-based method in (14) with entropy as the measure of uncertainty and cosine similarity as the measure of similarity. The value of the tuning parameter is set to a fixed value of $\beta_t = 1$.

In the simulation experiments, we randomly draw 50 instances from the training data, with 25 from each class, to form L_1 for each dataset. All models with an active learning component are initially trained using L_1, and an instance is selected at each iteration until the stopping criterion is met. We partition the instances in L_1 equally into ten folds with five instances in each fold. This same partition is applied to all stacking-related models, including the STACK model. Specifically, the first selected instance is added to the first fold, and the subsequent instances are added sequentially to subsequent folds.

In terms of performance evaluation, we consider two metrics: Accuracy and Area Under the Receiver Operating Characteristic (ROC) Curve (AUC). The accuracy is defined as $1 - \text{err}$, where err is defined in Section 2. The ROC curve illustrates the trade-off between the true positive rate (TPR) and the false positive rate (FPR) at various threshold settings. The AUC is the area under the ROC curve and provides a single scalar value that summarizes the classifier's overall performance across all possible threshold settings. At each iteration, the numerical performances of each model are evaluated on the test set.

In this simulation, we aim to demonstrate the numerical performances of active learning algorithms in the following aspects:

1. Comparison of applying the same active learning selection criterion to the weak learner, the boosted learner and the stacked learner. For instance, SVM-ENTROPY vs. ELTALB-ENTROPY vs. ELTALS-ENTROPY.
2. Comparison of different SVM-related and BOOST-related active learning algorithms with the vanilla SVM model, i.e., SVM-UNIF vs. SVM-ENTROPY vs. ELTALB-UNIF vs. ELTALB-ENTROPY vs. ALDEL-EELM.
3. Comparison of different stacking-related active learning algorithms with the vanilla stacking model, i.e., STACK vs. ELTALS-UNIF vs. ELTALS-ENTROPY vs. ALDELS-ENTROPY1 vs. ALDEL-ENTROPY2 vs. ALDELS-QBC vs. ALDELS-DENSITY.

In Figure 7, the accuracy of SVM, BOOST, and STACK models trained on both the full dataset and labeled subsets selected by ENTROPY algorithms at different stages are displayed using dashed and solid lines, respectively. These accuracies were averaged across all $S = 100$ simulations. Concerning the full models, it is evident that both BOOST and STACK models display higher accuracies compared to SVM. Specifically, the accuracy of the STACK model surpasses SVM by an impressive 13%, whereas the BOOST model has a relatively marginal improvement of approximately 1%. As for active learning algorithms, the ranking of the three metric curves remains unaltered compared to those of the curves from full models. However, the accuracies of active learning algorithms are noticeably lower than those of the full models, and the differences between them become less significant as the algorithms select more instances.

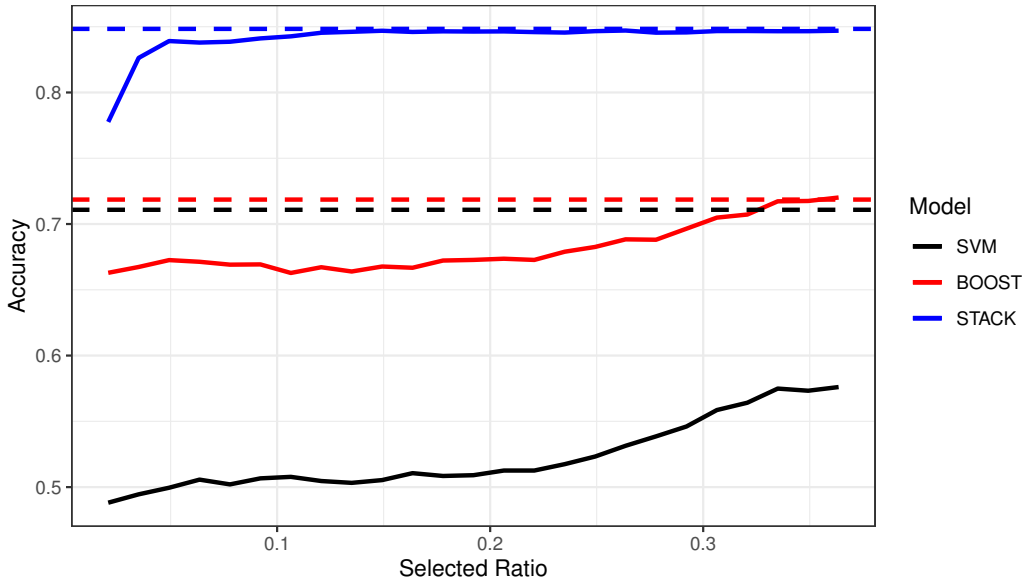

Figure 7. A comparison of the average Accuracy for SVM, BOOST and STACK models trained on both full data and labeled subsets obtained through the ENTROPY active learning algorithms. The classification metrics are assessed on the test datasets at various stages of instance selection. Results for SVM, BOOST, and STACK are depicted with dashed lines, while solid lines represent the performances of the corresponding ENTROPY-variant models (i.e., SVM-ENTROPY, BOOST-ENTROPY, and ELTALB-ENTROPY, respectively).

Figure 8 presents the results of a comparative analysis of active learning algorithms applied to the SVM and the BOOST models, aimed at investigating the impact of boosting and active learning simultaneously. The figure provides insights into the efficacy of different active learning strategies, with the ENTROPY algorithm outperforming random sampling for both the SVM and the BOOST models, showcasing the potential of active learning. Furthermore, the ELTALB-UNIF and ELTALB-ENTROPY algorithms, applied to the BOOST model, exhibit superior performance compared to the SVM-UNIF and SVM-ENTROPY algorithms, highlighting the power of boosting. ALDELB-EELM, acting as a bridge between the SVM and BOOST models, demonstrates a performance level that falls between the BOOST-related and the SVM-related active learning algorithms when only a few instances are labeled and trained. These findings provide insights into the non-interfering improvement of SVM models from both active learning and Boosting. Boosting is effective for both models from the full and partial data selected by an active learning algorithm. On the other hand, an active learning algorithm can select more informative instances for labeling for both unboosted and boosted models.

Figure 9 illustrates the effectiveness of different active learning algorithms in the context of STACK. Their performances are assessed on test data using average accuracy and AUC metrics. The results show that ELTALS-UNIF significantly underperforms compared to other algorithms. In terms of AUC, ALDELS-DENSITY is more effective, surpassing the performance of both ALDEL and ELTAL algorithms. However, there is no marked performance difference among the other algorithms.

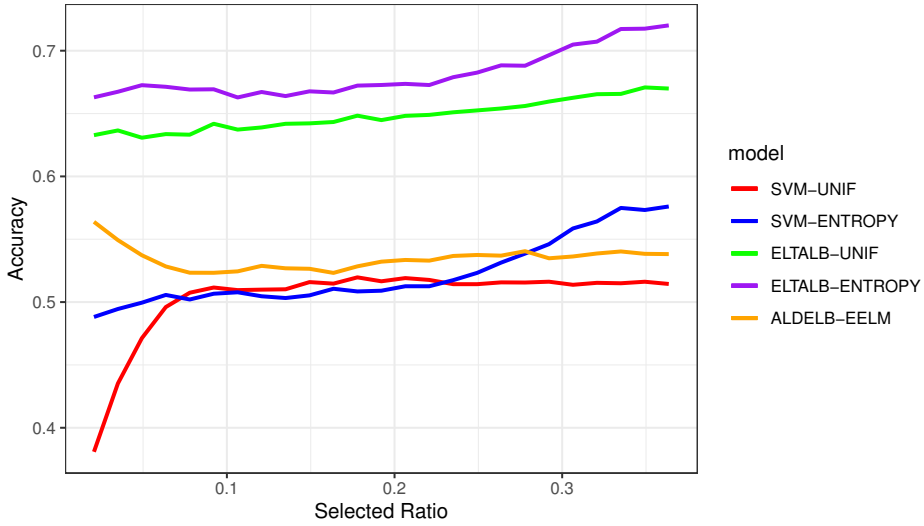

Figure 8. A comparison of the average Accuracy and AUC for SVM models trained on the labeled subsets selected by various SVM-related or Boosting-related active learning algorithms. The classification metrics are assessed on the test datasets at various stages of instance selection.

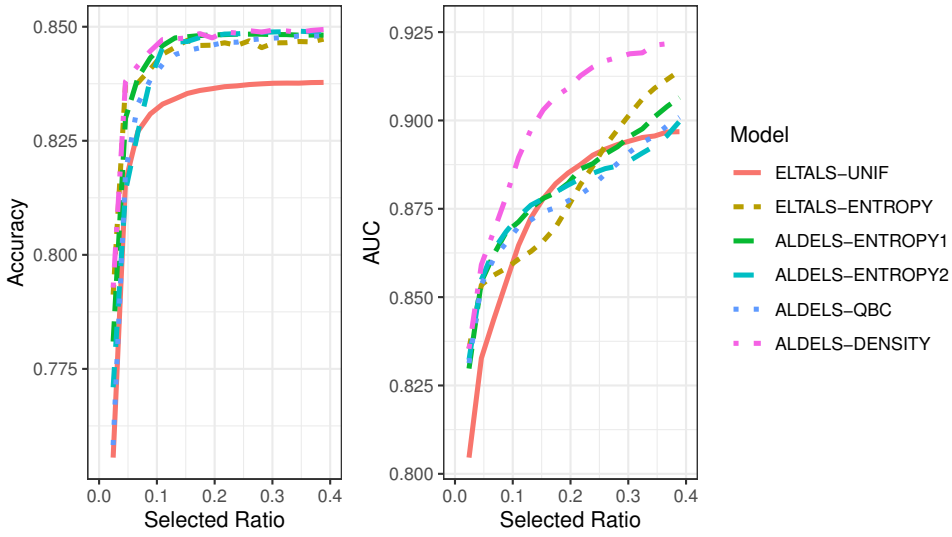

Figure 9. A comparison of the average Accuracy and AUC for STACK models trained on the labeled subsets selected by various STACK-related active learning algorithms. The classification metrics are assessed on the test datasets at various stages of instance selection.

Table 2 presents a detailed summary of the outcomes obtained from the complete SVM, BOOST, and STACK models, in conjunction with the corresponding active learning techniques introduced in Table 1. The row for each full model delineates the time elapsed in seconds during training when 5%, 10%, 25% and 50% of all instances are used and the accuracy assessed on the test data using all available training data. Conversely, the row for each active learning algorithm describes the duration of selecting an unlabeled instance, as well as the improvement and deterioration in accuracy measured on the test data relative to the UNIF active learning algorithm and the full model at each stage of the selection process, respectively. The test data accuracy for the SVM model is reported as 71%, which is surpassed by the BOOST and STACK models, registering accuracies of 72% and 85%, respectively. When considering the training time, the SVM model exhibits a rapid training time of 0.04 s when trained on half of the available data. In comparison, the BOOST and STACK models require a longer training duration of 2.35 s and 4.29 s, respectively.

The results from Table 2 demonstrate that the improvement in accuracy, as compared to SVM-UNIF, ELTALB-UNIF, and ELTALS-UNIF models, is consistently positive, indicating that the active learning algorithms employed are effective at all stages of selection.

At a selection ratio of 5%, ALDELB-EELM demonstrates a 13% accuracy improvement over SVM-UNIF, while experiencing a 16.5% reduction in accuracy relative to the full SVM model. It also has a significant improvement over SVM-ENTROPY up until the point where half of the instances are selected for use. The concurrent process of boosting and selection leads to a noteworthy reduction in the time required for instance selection (0.028 s compared to 0.669 s at a selection ratio of 5%). However, this approach also brings a reduction in accuracy of approximately 10% when 5% of instances are selected and about 17% when half of the instances are used.

Among the active learning algorithms associated with the STACK model, ALDELS-DENSITY exhibits a markedly longer selection time compared to its competitors. This is due to the computation of the similarity matrix for the unlabeled dataset at each iteration, as outlined in (14). The duration of this process becomes shorter when there are fewer unlabeled instances at a later stage. On the other hand, its performance is superior to other STACK-related AL algorithms. At a selection ratio of 5%, ALDELS-DENSITY incurs only a 3.5% additional error compared to the full STACK model, whereas other active learning algorithms show error increases ranging from 3.7% to 6.6%. The time taken to select data using ALDELS-ENTROPY1 and ALDELS-ENTROPY2 is longer compared to ELTALS-ENTROPY and ALDELS-QBC. This increased duration is because both ALDELS-ENTROPY1 and ALDELS-ENTROPY2 necessitate retraining of models, either at the lower or upper level, during their second-level selection process. ELTALS-ENTROPY boasts the shortest selection duration among all active learning algorithms linked to the STACK model. However, its selection process takes place only after the complete training of the STACK model, in contrast to other algorithms which require only the training of the lower-level models.

Table 2. Numerical Summaries of all SVM, BOOST and STACK-related active learning algorithms compared to the full model and the corresponding random sampling active learning algorithm in terms of average time elapsed during querying, training and the average improvement of Accuracy for the first 5%, 10%, 25%, 50% instances selected.

	5%		**10%**		**25%**		**50%**	
	Time	**vs. UNIF vs. Full**	**Time**	**vs. UNIF vs. Full**	**Time**	**vs. UNIF vs. Full**	**Time**	**vs. UNIF vs. Full**
SVM	0.006	0.711	0.007	0.711	0.016	0.711	0.040	0.711
SVM-ENTROPY	0.017	0.071 −0.218	0.019	0.032 −0.211	0.028	0.009 −0.204	0.043	0.02 −0.180
ALDELB-EELM	0.028	0.130 −0.165	0.031	0.069 −0.181	0.047	0.035 −0.185	0.084	0.031 −0.181
BOOST	0.332	0.719	0.420	0.719	0.924	0.719	2.349	0.719
ELTALB-ENTROPY	0.669	0.032 −0.052	0.809	0.034 −0.049	1.327	0.029 −0.048	2.042	0.042 −0.020
STACK	2.162	0.848	2.220	0.848	2.797	0.848	4.287	0.848
ELTALS-ENTROPY	0.596	0.030 −0.037	0.633	0.019 −0.021	0.757	0.014 −0.009	0.892	0.011 −0.005
ALDELS-ENTROPY1	1.375	0.021 −0.047	1.451	0.016 −0.024	1.877	0.013 −0.009	2.689	0.012 −0.004
ALDELS-ENTROPY2	1.703	0.008 −0.059	1.806	0.005 −0.035	2.242	0.009 −0.013	3.245	0.010 −0.006
ALDELS-QBC	0.599	0.002 −0.066	0.644	0.005 −0.035	0.749	0.008 −0.015	0.878	0.009 −0.007
ALDELS-DENSITY	44.16	0.033 −0.035	41.97	0.022 −0.018	39.01	0.016 −0.007	35.04	0.013 −0.003

Table 2 demonstrates the mutual benefits of active learning and ensemble learning in enhancing a basic machine learning model. On one hand, active learning can significantly reduce the labeling and computational costs of an ensemble learning model. Despite using only a limited number of labeled instances, active learning can achieve efficiency comparable to that of the ensemble learning model that labels all instances. In other words, active learning addresses the primary computational weakness of ensemble learning models. On the other hand, ensemble learning can further improve the accuracy of a model generated by an active learning algorithm. While the core idea of active learning is to trade off some efficiency for lower labeling and computational costs, by combining with ensemble learning, active learning can create a model that requires labeling only a small number of instances but outperforms the basic model that labels all instances. For example, the SVM model in Table 2 has an accuracy of 71.1%. However, by labeling only 10% of the instances and with a negligible selection time of 0.64 s, ALDELS-QBC can build a stacking model with an accuracy of 78.2%.

6.2. Real Data Application

In this Section, we illustrate the performances of different active learning algorithms by two real data applications: fitness exercise pose classification and smoke detection.

The fitness exercise poses classification dataset includes physical poses from 500 Youtube video clips of people exercising. The fitness exercise dataset analyzed in this subsection is publicly available in the Kaggle repository, https://www.kaggle.com/datasets/muhannadtuameh/exercise-recognition, accessed on 27 February 2023. These videos are part of Countix, which is a real-world dataset that is focused on repeated actions. The covariates in the dataset are extracted from the MediaPipe framework which takes the video as an input and predicts the location of 33 pose landmarks such as left eye inner, left eye outer, left heel, etc. All instances come from 10 physical poses that can be further integrated into five different exercises. The accurate classification of the fitness exercises requires expertise in fitness training and thus we hope to apply active learning to mitigate the cost of manual labeling. We only keep the 296 cases from the jumping-jack and 231 from the pull-up classes for binary classification and there are 100 covariates in total.

The smoke detection dataset comprises numeric recordings of various indoor conditions and gas concentrations. Based on this, the detector decides whether it is alarming or not and is treated as the response variable. This dataset is publicly available in the Kaggle repository, https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset, accessed on 28 February 2023. The original dataset has 60,000 observations and 13 covariates. Due to the imbalances between the two classes, 2500 instances are selected from each class to construct the dataset and all covariates are standardized.

For both real-world applications, 80% of instances are picked for training data, leaving the remaining to form the test data for measuring the classification models' performances. The models are put to the test in $S = 100$ simulations of varying initializations, and all are trained on the same 50 instances, 25 from each class. We implement all models using the full dataset and all active learning algorithms introduced in Table 1 then feverishly query and label instances in sequence until the training dataset is depleted.

Tables 3 and 4 show the average and standard error of Accuracy and F1-score metrics for all the active learning algorithms listed in Table 1 on the fitness exercise and smoke detection datasets, respectively. They illustrate how these algorithms perform with varying proportions of training data selected for labeling. The optimal non-stacking-related AL algorithm is highlighted in **bold red**, whereas the best stacking-related AL algorithm is represented in **bold blue** for each selection ratio.

In Table 3, the full SVM model achieves an accuracy of 80.2% on the test set, which is considerably lower than the boosted model's accuracy of 96.23% and the stacking model's accuracy of 94.08%. During the selection process, the SVM model with random selection performs better than the ENTROPY variant in uncertainty sampling. ELTALB-ENTROPY stands out by outperforming all other algorithms in terms of both accuracy and F1-score,

which showcases the strength of active learning combined with boosting techniques. Although ALDELB-EELM does not surpass other boosting-related algorithms in terms of performance, it still exhibits superior accuracy and F1-score compared to the SVM-related active learning algorithms. As for stacking-related algorithms, all proposed active learning algorithms are superior to ELTALS-UNIF at all selection stages in terms of both accuracy and F1-score. Among these, ELTALS-ENTROPY becomes the winner among the rest, while ALDELS-DENSITY has a relative lower standard error but also a slightly lower averaged measurement. However, the selection of ALDELS-DENSITY does not depend on the training of upper-level, which is a plus for the algorithm compared to ELTALS-ENTROPY.

Table 3. A summary of the mean and standard error of the accuracy and F1 score measured on the test set of the fitness exercise dataset when different sizes of instances are labeled and trained. Results from all full models and the sub models trained using the labeled subsets selected by different active learning algorithms are displayed. The accuracy and F-1 score are measured for $S = 100$ times with different initializations.

	Accuracy				**F1 Score**			
Selected Ratio	**20%**	**30%**	**50%**	**100%**	**20%**	**30%**	**50%**	**100%**
SVM-UNIF	79.64 (0.24)	80.17 (0.23)	79.61 (0.17)	80.20	76.68 (0.43)	77.78 (0.45)	77.36 (0.29)	77.89
SVM-ENTROPY	75.50 (1.46)	70.16 (2.22)	73.42 (1.01)	80.20	72.82 (2.76)	60.26 (5.29)	73.00 (1.98)	77.89
ELTALB-UNIF	85.16 (0.54)	87.33 (0.63)	87.91 (0.85)	96.23	82.95 (1.45)	86.11 (1.61)	87.36 (2.00)	96.79
ELTALB-ENTROPY	**89.60 (0.62)**	**91.97 (1.01)**	**93.51 (0.61)**	96.23	**87.60 (1.93)**	**91.64 (2.07)**	**93.75 (1.09)**	96.79
ALDELB-EELM	75.73 (1.72)	80.10 (1.61)	79.92 (1.66)	80.20	77.58 (2.89)	81.60 (3.09)	62.80 (3.73)	77.89
ELTALS-UNIF	80.45 (0.68)	83.49 (0.53)	85.86 (0.69)	94.08	77.04 (1.14)	80.00 (1.31)	83.94 (1.12)	93.38
ELTALS-ENTROPY	**87.01 (0.68)**	**90.94 (0.70)**	**92.80 (0.50)**	94.08	**87.35 (0.62)**	**90.13 (1.19)**	**92.51 (0.13)**	93.38
ALDELS-ENTROPY1	82.05 (0.63)	87.62 (1.11)	91.31 (0.89)	94.08	82.14 (2.11)	88.47 (1.23)	90.99 (1.17)	93.38
ALDELS-ENTROPY2	82.15 (1.27)	86.97 (1.34)	90.87 (0.90)	94.08	84.73 (1.12)	87.08 (1.85)	90.19 (1.63)	93.38
ALDELS-QBC	80.73 (1.00)	86.30 (0.62)	90.83 (0.42)	94.08	78.96 (1.92)	84.11 (1.50)	89.54 (0.56)	93.38
ALDELS-DENSITY	86.31 (0.33)	90.79 (0.23)	92.02 (0.28)	94.08	85.04 (0.58)	89.90 (0.38)	91.73 (0.43)	93.38

In Table 4, the full stacking model has a staggering 99.7% of accuracy and F1-score, which makes it more superior to the boosting model. With an approximate 90% of accuracy and F1-score, SVM still remains the worst full model. Here ELTALB-ENTROPY is still the best overall algorithm among all the SVM and boosting-related algorithms although the difference between ELTALB-UNIF and ELTALB-ENTROPY is no longer evident. For stacking-related algorithms, ALDELS-ENTROPY1, ALDELS-ENTROPY2 and ELTALS-ENTROPY all have a similar performance and those algorithms can create a stacking model that has 99.8% of accuracy when a merely 5% of instances are labeled and trained. The standard error of those three algorithms' accuracy and F1-score remain as low as 0.01%, which makes them even more competitive.

Based on the analysis of real data applications, both the stacking and boosting approaches outperform the full SVM model. Throughout the selection stages, ELTALB-ENTROPY almost consistently proves to be the most competitive among all SVM and boosting-related models. Regarding stacking-related algorithms, each active learning algo-

rithm demonstrates greater efficiency than the full SVM model, even when a small number of instances are selected. These findings motivate the adoption of active learning algorithms in ensemble learning models with a limited number of labeled instances instead of utilizing all the data and constructing a basic machine learning model. It is worth noting that while ALDELS-ENTROPY1 and ALDELS-ENTROPY2 employ different selection criteria based on loss at different levels, their performances are quite similar. Although it is challenging to pick a single preferred stacking-related active learning algorithm among the others, all of them exhibit significant improvements compared to random selection.

Table 4. A summary of the mean and standard error of the accuracy and F1 score measured on the test set of the smoke detection dataset when different sizes of instances are labeled and trained. Results from all full models and the sub models trained using the labeled subsets selected by different active learning algorithms are displayed. The accuracy and F-1 are measured for $S = 100$ times with different initializations.

	Accuracy				**F1 Score**			
Selected Ratio	**5%**	**10%**	**25%**	**100%**	**5%**	**10%**	**25%**	**100%**
SVM-UNIF	88.66 (0.28)	88.94 (0.28)	89.64 (0.17)	89.79	**89.21 (0.27)**	89.50 (0.29)	90.18 (0.19)	90.26
SVM-ENTROPY	83.99 (3.05)	89.36 (1.21)	86.88 (1.52)	89.79	83.41 (3.16)	89.42 (1.57)	85.80 (2.31)	90.26
ELTALB-UNIF	89.03 (2.15)	91.23 (1.43)	**92.87 (0.72)**	95.05	86.17 (3.63)	91.87 (2.04)	**93.49 (0.50)**	95.25
ELTALB-ENTROPY	**89.34 (2.95)**	**94.66 (1.98)**	92.57 (2.32)	95.05	85.43 (4.31)	**94.45 (2.34)**	90.87 (3.29)	95.25
ALDELB-EELM	71.78 (3.47)	65.19 (2.92)	59.75 (2.85)	89.79	63.61 (5.12)	65.47 (3.72)	62.85 (3.95)	90.26
ELTALS-UNIF	98.60 (0.10)	99.05 (0.06)	99.53 (0.04)	99.70	98.69 (0.10)	99.15 (0.05)	99.58 (0.03)	99.70
ELTALS-ENTROPY	99.87 (0.01)	99.71 (0.01)	99.71 (0.01)	99.70	99.86 (0.01)	99.71 (0.01)	99.71 (0.01)	99.70
ALDELS-ENTROPY1	99.85 (0.02)	**99.77 (0.02)**	99.69 (0.01)	99.70	**99.88 (0.01)**	**99.76 (0.01)**	99.69 (0.01)	99.70
ALDELS-ENTROPY2	**99.87 (0.01)**	99.72 (0.01)	**99.71 (0.01)**	99.70	99.87 (0.01)	99.73 (0.01)	**99.71 (0.01)**	99.70
ALDELS-QBC	97.78 (0.26)	98.15 (0.17)	98.16 (0.22)	99.70	99.73 (0.27)	98.13 (0.15)	98.15 (0.20)	99.70
ALDELS-DENSITY	99.51 (0.23)	99.47 (0.18)	99.52 (0.13)	99.70	99.53 (0.22)	99.48 (0.17)	99.53 (0.12)	99.70

7. Discussion and Future Directions

This article presents a comprehensive framework for incorporating active learning into AdaBoost and stacking models. We introduce two distinct directions, namely ELTAL and ALDEL, which integrate active learning with ensemble learning based on the timing of instance selection. For each direction, we propose generalized algorithms applicable to both boosting and stacking.

In ELTAL algorithms, we provide examples using uncertainty sampling and random sampling to illustrate their application. In ALDEL for boosting, we introduce a query strategy that aims to select instances maximizing the expected exponential loss for labeling. The selected instance is then incorporated into the current labeled dataset for further boosting and training. In ALDEL for stacking, we present four active learning algorithms categorized as direct selection and two-level selection. The two-level selection allows individual lower-level models to independently select instances, followed by a second-level selection from the candidate instances. The direct selection treats lower-level training as

a transformation of the input data and applies existing active learning algorithms to the probabilistic output of the lower-level models.

We evaluate all proposed algorithms against basic support vector machine model using simulated datasets and two real-world applications. The results demonstrate the significant performance enhancement achieved by ensemble learning, whether utilizing full data or partially labeled data from active learning algorithms. Moreover, we find that even a sub ensemble learning model trained on a small number of instances selected by an active learning algorithm can outperform a support vector machine trained on the entire dataset. The slight decrease in performance compared to the ensemble learning model trained on all instances is negligible.

There are certain limitations in this work that warrant further investigation. Firstly, the proposed active learning algorithms lack a solid theoretical foundation. While the success of AdaBoost is supported by the minimization of exponential loss and the conversion from weak learners to strong learners, the introduction of ALDEL for Boosting and the related query strategy are derived from intuition and demonstrated through numerical examples. Secondly, it is important to explore the optimization of ALDEL for Stacking. While the two-level selection allows each lower-level model to select their preferred instances independently, the independent selection process could be further enhanced by considering the most suitable query strategy for each model. This can draw insights from a wide range of active learning research conducted for different models. In our numerical illustrations, all lower-level selections are based on uncertainty sampling using cross-entropy. However, exploring different query strategies and determining their efficacy for specific real-world applications would be valuable. Currently, we have not provided sufficient intuition on when to utilize each query strategy or the rationale behind a query strategy being more competitive for a particular application.

To enhance our grasp and use of active learning algorithms within ensemble learning models, it's crucial to overcome current limitations. Future research should concentrate on three key areas: providing theoretical foundations, optimizing active learning for Stacking methods, and determining the most effective query strategies for particular use cases. While we've laid out a basic framework for integrating active learning into ensemble learning, there's a need for more detailed studies and advancements in optimization techniques in upcoming research.

Author Contributions: Conceptualization, Q.S.; methodology, Q.S.; software, Q.S.; validation, Q.S.; formal analysis, Q.S.; investigation, Q.S.; resources, Q.S.; data curation, Q.S.; writing—original draft preparation, Q.S.; writing—review and editing, S.K.G.; visualization, Q.S.; supervision, S.K.G.; project administration, Q.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: This study utilized two publicly available datasets obtained from Kaggle, an online community of data scientists and machine learning practitioners. The datasets used are exercise recognition and smoke detection. Prior to their inclusion in this study, the datasets were reviewed for any ethical concerns. The datasets on Kaggle are provided under specific licenses that permit their free and open use for analytical and scientific research. The licenses associated with these datasets do not require individual consent as the data has been collected and processed in a manner consistent with Kaggle's terms of service, which ensure that data providers have obtained all necessary consents and permissions for data collection and sharing.

Data Availability Statement: The data that were used in this article are publicly available in the Kaggle repository at https://www.kaggle.com/datasets/muhannadtuameh/exercise-recognition, accessed on 27 February 2023. and https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset, accessed on 28 February 2023.

Conflicts of Interest: The authors declare that they have no competing interests.

References

1. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1249. [CrossRef]
2. Freund, Y.; Schapire, R.E. Experiments with a new boosting algorithm. *ICML* **1996**, *96*, 148–156.
3. Freund, Y.; Schapire, R.E. Game theory, on-line prediction and boosting. In Proceedings of the Ninth Annual Conference on Computational Learning Theory, Desenzano del Garda, Italy, 28 June–1 July 1996; pp. 325–332.
4. Freund, Y.; Schapire, R.E. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. [CrossRef]
5. Friedman, J.H. Stochastic gradient boosting. *Comput. Stat. Data Anal.* **2002**, *38*, 367–378. [CrossRef]
6. Wolpert, D.H. Stacked generalization. *Neural Netw.* **1992**, *5*, 241–259. [CrossRef]
7. Breiman, L. Stacked regressions. *Mach. Learn.* **1996**, *24*, 49–64. [CrossRef]
8. Lewis, D.D.; Gale, W.A. A sequential algorithm for training text classifiers. In Proceedings of the SIGIR'94, Dublin, Ireland, 3–6 July 1994; pp. 3–12.
9. Settles, B. *Active Learning Literature Survey*; University of Wisconsin-Madison Department of Computer Sciences: Madison, WI, USA, 2009.
10. Roy, N.; McCallum, A. Toward optimal active learning through sampling estimation of error reduction. *Int. Conf. Mach. Learn.* **2001**, 441–448.
11. Müller, B.; Reinhardt, J.; Strickland, M.T. *Neural Networks: An Introduction*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1995.
12. Ren, P.; Xiao, Y.; Chang, X.; Huang, P.Y.; Li, Z.; Gupta, B.B.; Chen, X.; Wang, X. A survey of deep active learning. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–40. [CrossRef]
13. Abdar, M.; Pourpanah, F.; Hussain, S.; Rezazadegan, D.; Liu, L.; Ghavamzadeh, M.; Fieguth, P.; Cao, X.; Khosravi, A.; Acharya, U.R.; et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Inf. Fusion* **2021**, *76*, 243–297. [CrossRef]
14. Gal, Y.; Islam, R.; Ghahramani, Z. Deep bayesian active learning with image data. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 1183–1192.
15. 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 Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 9368–9377.
16. Sener, O.; Savarese, S. A geometric approach to active learning for convolutional neural networks. *arXiv* **2017**, arXiv:1708.00489.
17. Pop, R.; Fulop, P. Deep ensemble bayesian active learning: Addressing the mode collapse issue in monte carlo dropout via ensembles. *arXiv* **2018**, arXiv:1811.03897.
18. Valiant, L.G. A theory of the learnable. *Commun. ACM* **1984**, *27*, 1134–1142. [CrossRef]
19. Schapire, R.E. The strength of weak learnability. *Mach. Learn.* **1990**, *5*, 197–227. [CrossRef]
20. Zhang, T.; Yu, B. Boosting with early stopping: Convergence and consistency. *Ann. Stat.* **2005**, 1538. [CrossRef]
21. Mease, D.; Wyner, A. Evidence Contrary to the Statistical View of Boosting. *J. Mach. Learn. Res.* **2008**, *9*, 131–156.
22. Schapire, R.E.; Singer, Y. Improved boosting algorithms using confidence-rated predictions. In Proceedings of the Eleventh Annual Conference on Computational Learning Theory, Madison, WI, USA, 24–26 July 1998; pp. 80–91.
23. Friedman, J.; Hastie, T.; Tibshirani, R. Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors). *Ann. Stat.* **2000**, *28*, 337–407. [CrossRef]
24. Polikar, R. Ensemble learning. In *Ensemble Machine Learning: Methods and Applications*; Springer: New York, NY, USA, 2012; pp. 1–34.
25. Raftery, A.E.; Madigan, D.; Hoeting, J.A. Bayesian model averaging for linear regression models. *J. Am. Stat. Assoc.* **1997**, *92*, 179–191. [CrossRef]
26. Hoeting, J.A.; Madigan, D.; Raftery, A.E.; Volinsky, C.T. Bayesian model averaging: A tutorial (with comments by M. Clyde, David Draper and EI George, and a rejoinder by the authors. *Stat. Sci.* **1999**, *14*, 382–417. [CrossRef]
27. Wasserman, L. Bayesian model selection and model averaging. *J. Math. Psychol.* **2000**, *44*, 92–107. [CrossRef]
28. Clarke, B. Comparing Bayes model averaging and stacking when model approximation error cannot be ignored. *J. Mach. Learn. Res.* **2003**, *4*, 683–712.
29. Zhou, Z.H. *Machine Learning*; Springer Nature: Singapore, 2021.
30. Seung, H.S.; Opper, M.; Sompolinsky, H. Query by committee. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, PA, USA, 27–29 July 1992; pp. 287–294.
31. Kumar, P.; Gupta, A. Active learning query strategies for classification, regression, and clustering: A survey. *J. Comput. Sci. Technol.* **2020**, *35*, 913–945. [CrossRef]
32. Settles, B.; Craven, M.; Ray, S. Multiple-instance active learning. *Adv. Neural Inf. Process. Syst.* **2007**, *20*, 1289–1296.
33. Fu, Y.; Zhu, X.; Li, B. A survey on instance selection for active learning. *Knowl. Inf. Syst.* **2013**, *35*, 249–283. [CrossRef]
34. Wu, Y.; Kozintsev, I.; Bouguet, J.Y.; Dulong, C. Sampling strategies for active learning in personal photo retrieval. In Proceedings of the 2006 IEEE International Conference on Multimedia and Expo, Toronto, ON, Canada, 9–12 July 2006; pp. 529–532.
35. Li, X.; Guo, Y. Adaptive active learning for image classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 859–866.

36. Yang, Y.; Loog, M. A variance maximization criterion for active learning. *Pattern Recognit.* **2018**, *78*, 358–370. [CrossRef]
37. Wang, L.M.; Yuan, S.M.; Li, L.; Li, H.J. Boosting Naïve Bayes by active learning. In Proceedings of the 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826), Shanghai, China, 26–29 August 2004; Volume 3, pp. 1383–1386.
38. Abe, N. Query learning strategies using boosting and bagging. In Proceedings of the Fifteenth International Conference on Machine Learning, San Francisco, CA, USA, 24–27 July 1998; pp. 1–9.
39. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
40. Kullback, S.; Leibler, R.A. On information and sufficiency. *Ann. Math. Stat.* **1951**, *22*, 79–86. [CrossRef]
41. Menéndez, M.; Pardo, J.; Pardo, L.; Pardo, M. The jensen-shannon divergence. *J. Frankl. Inst.* **1997**, *334*, 307–318. [CrossRef]
42. Hearst, M.A.; Dumais, S.T.; Osuna, E.; Platt, J.; Scholkopf, B. Support vector machines. *IEEE Intell. Syst. Their Appl.* **1998**, *13*, 18–28. [CrossRef]
43. Platt, J. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Adv. Large Margin Classif.* **2000**, *10*, 61–74..
44. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
45. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
46. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]