APPENDIX

A RELATED WORK

We detail related work in unsupervised performance estimation here. Works below assume access to *only* unlabeled data; in contrast, SSME learns from both labeled and unlabeled data.

Unsupervised performance estimation involves estimating the performance of a model given only unlabeled data. Methods designed to address this problem often focus on out-of-distribution samples, where labeled data is scarce and model performance is known to degrade. Several works have illustrated strong empirical relationships between out-of-distribution generalization and thresholded classifier confidence (Garg et al., 2022), dataset characteristics (Deng & Zheng, 2021; Guillory et al., 2021), in-distribution classifier accuracy (Miller et al., 2021), and classifier agreement (Parisi et al., 2014; Platanios et al., 2017; Baek et al., 2022).

Several works have formalized when unsupervised model evaluation is possible (Donmez et al., 2010; Chen et al., 2022; Garg et al., 2022; Lu et al., 2023), and propose assumptions under which estimates of performance are recoverable. Donmez et al. (2010) and Balasubramanian et al. (2011) assume knowledge of $p(y)$ in the unlabeled sample. Steinhardt & Liang (2016) assume conditionally-independent subsets of the observed features, inspired by conditional-independence assumptions made in works such as Dawid & Skene (1979). Guillory et al. (2021) assume classifier calibration on unlabeled samples. Chen et al. (2022) assume a sparse covariate shift model, in which a subset of the features' class-conditional distribution remains constant. Lu et al. (2023) illustrate misestimation of $p(y)$ in the unlabeled example, and assume that $p(y)$ out-of-distribution is close to $p(y)$ in-distribution. As Garg et al. (2022) highlight, assumptions are necessary to make any claim about the nature of unsupervised model evaluation, and the above methods are a representative sample of assumptions made by prior works.

Our work is also similar, in spirit, to methods that learn to debias classifier predictions on a small set of labeled data and then apply that debiasing procedure to classifier predictions on unlabeled examples. Prediction-powered inference (Angelopoulos et al., 2023) and double machine learning (Chernozhukov et al., 2018) both learn a debiasing procedure to ensure that unlabeled metric estimates (e.g., accuracy) are statistically unbiased. One of the baselines we compare to, AutoEval (Boyeau et al., 2024), is built atop prediction-powered inference.

B EXPERIMENTAL DETAILS

B.1 REAL DATASETS AND CLASSIFIER SETS

We provide additional detail for the six datasets we use in our work, including ground truth $p(y)$ for each dataset and ground truth metrics for each classifier in the associated classifier set **[DS: ref]**. As discussed, each dataset is split into a training split (provided to each classifier as training data), an estimation split (provided to each performance estimation method), and an evaluation split (used to compute ground truth metrics for each classifier). We determine training splits based on prior work. We then split the remaining data in half (randomly, for each run) to produce the estimation and evaluation splits. We then ubsample the estimation split to have n_l labeled examples and n_u unlabeled examples. We ensure that the labeled data always includes at least one example from each class. Thus, the estimation split contains $n_l + n_u$ examples in each experiment, and the evaluation split for each task is fixed across runs (exact sample sizes reported below).

1. **MIMIC-IV**: We use three binary classification tasks from MIMIC-IV (Johnson et al., 2020), a large dataset of electronic health records describing 418K patient visits to an emergency department. We focus on three tasks: **hospitalization** (predicting hospital admission based on features available during triage, $p(y = 1) = 0.45$), **critical outcomes** (predicting inpatient mortality or a transfer to the ICU within 12 hours, $p(y = 1) = 0.06$), and **emergency department revisits** (predicting a patient's return to the emergency department within 3 days, $p(y = 1) = 0.03$). We split and preprocess data according to prior work (Xie et al., 2022; Movva et al., 2023). No patient appears in more than one split. For each task, the evaluation split contains 70,439 examples.

2. **Toxicity detection**: The task is to predict presence of toxicity given an online comment, using data from CivilComments (Borkan et al., 2019; Koh et al., 2021) where $p(y = 1) = 0.11$. The evaluation split contains 66,891 examples.
3. **Biochemical property prediction** The task is to predict presence of a biochemical property based on a molecular graph, using data from the Open Graph Benchmark (Hu et al., 2020). We focus on the task of predicting whether a molecule inhibits SARS-CoV virus maturation, where $p(y = 1) = 0.09$. We filter out examples for which *no* label is observed (i.e. the molecule was not screened at all) because it is impossible to evaluate our performance estimates on those examples. Doing so reduces data held-out from training from 43,793 to 28,325 examples. The evaluation split then contains half, or 14,163, of those examples.
4. **News classification** The task is to predict one of four news types based on the title and description of an article (Zhang et al., 2015). The classes are balanced and the evaluation split contains 3,800 examples.
5. **Sentence classification** The task is to predict one of three textual entailments from a sentence (Williams et al., 2018). The classes are balanced and the evaluation split contains 61,856 examples.
6. **Image classification** The task is to predict one of nine coarse image categories (e.g. "dog" or "vehicle") from an image (Xiao et al.). The classes are balanced and the evaluation split contains 2,025 examples.

B.2 Baselines

For baselines that require discrete predictions (i.e. Dawid-Skene and AutoEval), we discretize classifier scores by assigning a class according to the maximum classifier score across classes. We expand on our implementation of each baseline below.

- *Labeled*: When estimating performance over the whole dataset, we compare the classifier scores to the ground truth labels within the labeled sample. However, when estimating subgroup-specific performance, it is often the case that there are no labeled examples for a given subgroup. In these instances, *Labeled* reverts to estimating subgroup-specific performance as performance over all labeled examples.
- *Pseudo-Labeling*: We train a logistic regression with the default parameters associated with the scikit-learn implementation (Pedregosa et al., 2011). Experiments with alternative function classes (e.g. a KNN) revealed no significant differences in performance.
- *Bayesian-Calibration*: Bayesian-Calibration operates on each classifier individually. We make use of the implementation made available by Ji et al. (2020). Extending the proposed approach to multi-class tasks is not straightforward, so we compare to *Bayesian-Calibration* only on binary tasks.
- *Dawid-Skene*: We implement Dawid-Skene with a tolerance of 1e-5 and a maximum number of EM iterations of 100, according to a public implementation.
- *AutoEval*: We implement AutoEval using an implementation made available by the authors (Boyeau et al., 2024). The implementation, to the best of our knowledge, only supports accuracy estimation across a set of classifiers, so we limit our comparison to this metric.

B.3 Semisynthetic dataset and classifier sets

As with the real datasets, we produce three splits: a training split to learn the classifiers (50 examples), an estimation split for the performance estimation methods (20 labeled examples and 1000 unlabeled examples), and an evaluation split to measure ground truth values for each metric (10,000 examples). Each classifier is a logistic regression with default L2 regularization.

B.4 Computing effective sample size

In order to compute effective sample size, we produce 50 samples of labeled data for each increment of 5 between 10 labeled examples and 1000. We then compute the mean absolute metric estimation error of using labeled data alone, across all runs. The effective sample size of a given

Dataset	Classifier	Acc	ECE	AUC	AUPRC
Hospital Admission	DT-RandomForest-seed1	74.2	1.5	81.5	76.0
	MLP-ERM-seed2	74.4	1.4	81.7	76.7
	MLP-ERM-seed1	74.4	1.9	81.9	77.0
	MLP-ERM-seed0	74.5	2.4	82.0	77.0
	LR-LBFGS-seed2	73.3	4.0	80.7	75.5
	LR-LBFGS-seed1	73.3	4.0	80.7	75.5
	LR-LBFGS-seed0	73.4	2.9	81.0	75.7
	DT-RandomForest-seed2	74.3	1.6	81.5	76.1
	DT-RandomForest-seed0	74.1	1.5	81.5	76.1
Critical Outcome	MLP-ERM-seed2	93.9	0.9	87.9	38.6
	MLP-ERM-seed1	93.9	0.8	88.1	39.0
	LR-LBFGS-seed2	93.6	1.2	87.6	34.2
	MLP-ERM-seed0	93.9	0.5	87.5	37.8
	LR-LBFGS-seed0	93.6	1.2	87.6	34.1
	DT-RandomForest-seed2	94.0	0.3	87.2	38.2
	DT-RandomForest-seed1	94.0	0.4	87.4	38.3
	DT-RandomForest-seed0	94.0	0.4	87.4	38.3
	LR-LBFGS-seed1	93.6	1.2	87.6	34.2
ED Revisit	DT-RandomForest-seed0	97.7	1.8	54.9	2.7
	DT-RandomForest-seed1	97.7	1.7	55.3	2.7
	DT-RandomForest-seed2	97.7	1.8	54.9	2.7
	LR-LBFGS-seed0	97.7	0.4	59.3	3.0
	LR-LBFGS-seed2	97.7	0.4	59.1	3.0
	MLP-ERM-seed0	97.7	0.3	59.8	3.1
	MLP-ERM-seed1	97.7	0.3	59.8	3.1
	MLP-ERM-seed2	97.7	0.5	57.9	3.0
	LR-LBFGS-seed1	97.7	0.4	59.1	3.0
Toxicity Detection	distilbert-CORAL-seed0	88.3	6.0	86.2	40.0
	distilbert-IRM-seed2	88.7	10.2	91.9	65.5
	distilbert-IRM-seed1	89.0	9.8	91.0	66.5
	distilbert-IRM-seed0	88.1	10.6	91.6	65.9
	distilbert-ERM-seed2	92.1	4.9	94.1	73.3
	distilbert-ERM-seed1	92.2	6.2	93.8	72.3
	distilbert-ERM-seed0	92.2	6.1	93.8	72.2
Molecule Property 60	gin-virtual-CORAL-seed1	92.8	5.2	90.1	61.9
	gin-virtual-CORAL-seed2	92.8	5.2	90.1	61.9
	gin-virtual-ERM-seed0	94.6	1.2	94.5	73.5
	gin-virtual-ERM-seed1	92.4	5.6	90.7	61.1
	gin-virtual-ERM-seed2	92.8	5.2	90.1	61.9
	gin-virtual-IRM-seed0	93.2	1.8	90.2	58.4
	gin-virtual-IRM-seed1	91.1	5.2	83.8	43.8
	gin-virtual-IRM-seed2	91.1	5.7	82.8	44.7

Table S1: **Ground truth classifier metrics on binary tasks.** We report ground truth performance for classifiers in the sets associated with each binary task. Each classifier name begins with the architecture (e.g. DT represents DecisionTree), the loss or training procedure (e.g. ERM or IRM), and then the seed. Note that the equivalent accuracies on ED Revisit are a byproduct of both the low class prevalence and the poor classifiers.

Dataset	Classifier	Acc	ECE
AG News	all-MiniLM-L12-v2	84.8	4.2
	mxbai-embed-large-v1	85.0	14.4
	multi-qa-MiniLM-L6-cos-v1	85.6	5.2
	bge-small-en-v1.5	85.2	16.9
	bge-large-en-v1.5	86.8	4.8
	bge-base-en-v1.5	86.6	5.6
	all-mpnet-base-v2	86.7	2.9
	all-MiniLM-L6-v2	83.8	3.8
	paraphrase-multilingual-MiniLM-L12-v2	85.1	9.6
	paraphrase-MiniLM-L6-v2	86.0	8.9
MultiNLI	distilbert-SqrtReWeight	81.4	9.2
	distilbert-ReWeight	80.9	7.4
	distilbert-ReSample	81.4	8.2
	distilbert-IRM	64.8	6.1
ImagenetBG	ResNet-ReWeight	86.6	7.8
	ResNet-ReSample	87.4	7.7
	ResNet-Mixup	88.6	7.7
	ResNet-IRM	54.1	30.9

Table S2: **Ground truth classifier metrics on multiclass tasks.** We report ground truth performance for classifiers in the sets associated with each multiclass task. Each of the LLMs fine-tuned for AG News are sentence transformers, while the MultiNLI classifiers all use DistilBERT (Sanh, 2019) as the base architecture. The base architecture on ImagenetBG is a ResNet-50.

semi-supervised evaluation method is thus the amount of labeled data which achieves the most similar mean absolute metric estimation error.

C Normalizing Flow

One alternative parameterization is to use a normalizing flow to model our mixture of distributions. Normalizing flows learn and apply an invertible transform f_θ to a random variable $\mathbf{z} \sim D_1$ to obtain $f_\theta(\mathbf{z}) \sim D_2$. Here, we set $\mathbf{z} \sim D_1$ to a Gaussian mixture model and learn a transformation such that $f_\theta(\mathbf{z}) \overset{\text{dist.}}{\approx} \mathbf{s}$, i.e., the transformed distribution roughly matches our classifier score distribution. By modeling $\mathbf{z}$ explicitly as a Gaussian mixture model, one can move back and forth between the two distributions, as $f_\theta^{-1}(f_\theta(\mathbf{z})) = \mathbf{z}$. Specifically, we set the distribution of $\mathbf{Z}$ to follow a Gaussian mixture:

$$\mathbf{Z}|(Y = k) \sim \mathcal{N}(\mu_k, \Sigma_k)$$

Thus, the marginal distribution of $\mathbf{Z}$ is $p_{\mathbf{Z}}(\mathbf{z}) = \sum_{k=1}^{K} \mathcal{N}(\mathbf{z}|\mu_k, \Sigma_k) \cdot p(y = k)$ is the overall density of $\mathbf{z}$. We apply our invertible transformation f_θ to obtain $\mathbf{s} = f_\theta(\mathbf{z})$. To find $p(\mathbf{s}|y = k)$, we follow the approach of Izmailov et al. (2020):

$$p_{\mathbf{S}}(\mathbf{s}|y = k) = \mathcal{N}(f_\theta^{-1}(\mathbf{s})|\mu_k, \Sigma_k) \cdot \left|\det\left(\frac{\delta f}{\delta x}\right)\right| \cdot p(y = k)$$

Intuitively, we transform $(\mathbf{s}, y)$ into a distribution $(\mathbf{z}, y)$ which follows a Gaussian mixture model. By enforcing the constraint that this transform is invertible, the joint distribution on $(\mathbf{z}, y)$ captures all the information in $(\mathbf{s}, y)$.

We use the RealNVP architecture (Dinh et al., 2016) to parameterize f_θ using 10 coupling layers, 3 fully-connected layers, and a hidden dimension of 128 between the fully connected layers. Our normalizing flow is lightweight and trains in less than a minute for each dataset in our experiments section using 1 80GB NVIDIA A100 GPU.

Note there are two optimizations here: (1) the normalizing flow transformation f_θ which maps $\mathbf{s}$ into our latent Gaussian mixture space and (2) the Gaussian mixture model parameters μ_k, Σ_k themselves. We begin by fixing the GMM parameters μ_k, Σ_k to values estimated from our classifier scores $\mathbf{s}$ and learning only the flow f_θ for 300 epochs. Afterwards, we optimize the GMM parameters μ_k, Σ_k with EM for another 700 epochs.

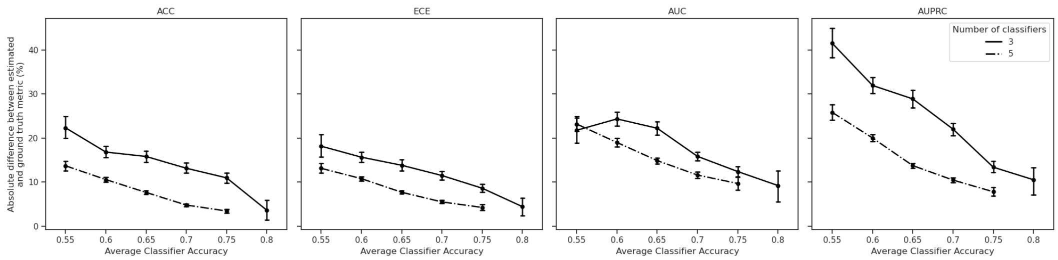

Figure S1: **Impact of average accuracy across classifiers in set on SSME's performance.**

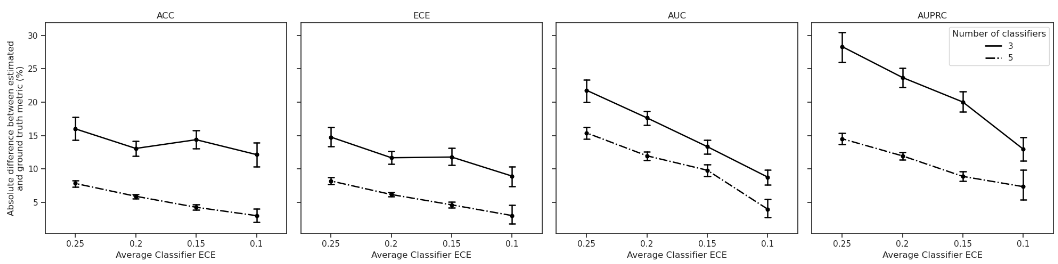

Figure S2: **Impact of average ECE across classifiers in set on SSME's performance.**

D Supplementary results

D.1 Results reporting mean absolute error

In the main text, we evaluate our method and all baselines using 20 labeled examples and 1000 unlabeled examples and report *rescaled* mean absolute error across metrics and tasks. Here, we supplement those results by reporting mean absolute error across each task and metric and expanding n_l to include 50 and 100. The number of unlabeled examples remains the same (1000) to isolate the effect of additional labeled data.

Tables S3, S4, S5, and S6 report our results on each binary task, for accuracy, ECE, AUC, and AUPRC , respectively. Three high-level findings emerge. First, SSME-KDE achieves the lowest mean absolute error (averaging across tasks and amounts of labeled data). Second, SSME-KDE consistently outperforms the ablated version of SSME, fit to a single model at a time (SSME-KDE-M). And finally, SSME-KDE is able to produce performance estimates that are quite close, in absolute terms, to ground truth. For example, when given 20 labeled examples and 1000 unlabeled examples, SSME-KDE estimates accuracy within at most 2.5 percentage points of ground truth accuracy (across tasks).

Tables S7 and S8 report our results on the multiclass tasks, for accuracy and ECE respectively. Note that we exclude Bayesian-Calibration from multiclass comparisons because the method does not natively support multiclass recalibration. We also omit AutoEval from Table S8 because the implementation of expected calibration error within the framework is not straightforward.

D.2 Comparison to ensembling

While we limit the scope of our experiments in the main text to semi-supervised methods that make use of *both* labeled and unlabeled data, another approach would be to produce an estimate of $Pr(y = k|s^{(i)})$ by averaging the classifier scores. This approach results in an unbiased metric estimator when the resulting ensemble is calibrated, as theoretical results by Ji et al. (2020) show. Such an approach has natural downsides: it is sensitive to the composition of the classifier set, does not improve with the introduction of labeled data, and relies on an assumption of ensemble calibration that is unlikely to hold in practice (Wu & Gales, 2021). Here, we provide experiments to illustrate this behavior.

Dataset	n_ℓ	n_u	Labeled	Pseudo-Labeling (LR)	Dawid-Skene	AutoEval	Bayesian-Calibration	SSME-KDE-M	SSME-KDE (Ours)
Critical Outcome	20	1000	5.19 ± 3.85	4.12 ± 3.87	4.36 ± 0.31	4.78 ± 3.34	2.80 ± 2.23	<u>1.70 ± 0.99</u>	**0.67 ± 0.46**
	50	1000	2.90 ± 2.13	3.06 ± 2.32	4.07 ± 0.40	3.01 ± 2.36	2.07 ± 1.29	<u>1.65 ± 0.90</u>	**0.78 ± 0.47**
	100	1000	2.09 ± 1.47	1.58 ± 1.08	3.87 ± 0.38	2.00 ± 1.16	<u>1.18 ± 0.74</u>	1.30 ± 0.70	**0.77 ± 0.47**
ED Revisit	20	1000	5.11 ± 3.53	5.13 ± 3.23	4.02 ± 2.83	4.70 ± 3.32	4.36 ± 2.76	<u>1.64 ± 1.24</u>	**0.45 ± 0.36**
	50	1000	2.02 ± 2.08	2.73 ± 2.24	2.74 ± 2.22	1.95 ± 2.07	2.47 ± 2.07	<u>1.46 ± 0.97</u>	**0.53 ± 0.39**
	100	1000	1.43 ± 1.15	1.54 ± 1.22	1.51 ± 1.18	1.42 ± 1.04	1.43 ± 1.12	<u>1.18 ± 0.89</u>	**0.57 ± 0.39**
Hospital Admission	20	1000	7.32 ± 4.52	6.86 ± 4.31	19.55 ± 0.47	7.19 ± 3.73	<u>2.48 ± 1.59</u>	3.29 ± 1.71	**1.88 ± 1.04**
	50	1000	5.40 ± 2.98	3.99 ± 2.97	18.78 ± 0.51	5.23 ± 2.46	<u>2.14 ± 1.28</u>	3.17 ± 1.85	**1.95 ± 0.99**
	100	1000	3.64 ± 1.99	3.01 ± 1.92	17.81 ± 0.59	4.01 ± 1.99	<u>2.42 ± 1.19</u>	3.06 ± 1.64	**1.51 ± 0.82**
SARS-CoV Inhibition	20	1000	6.11 ± 3.45	5.95 ± 3.62	4.91 ± 0.65	4.59 ± 3.05	**2.25 ± 1.13**	3.06 ± 0.83	<u>2.30 ± 0.56</u>
	50	1000	3.22 ± 2.05	2.99 ± 1.64	4.50 ± 0.63	2.64 ± 1.53	**1.74 ± 0.76**	2.59 ± 0.94	<u>2.35 ± 0.35</u>
	100	1000	2.04 ± 1.38	2.14 ± 1.10	4.01 ± 0.62	1.94 ± 0.90	**1.43 ± 0.68**	<u>1.84 ± 0.85</u>	2.36 ± 0.47
Toxicity Detection	20	1000	5.95 ± 2.64	5.03 ± 2.92	<u>4.82 ± 0.32</u>	5.27 ± 2.71	5.29 ± 1.06	6.71 ± 0.83	**2.34 ± 0.52**
	50	1000	4.03 ± 2.44	<u>2.88 ± 1.72</u>	4.65 ± 0.29	3.37 ± 1.48	4.57 ± 1.07	5.38 ± 1.01	**2.22 ± 0.47**
	100	1000	2.43 ± 1.48	**1.90 ± 1.11**	4.46 ± 0.40	2.34 ± 0.94	3.78 ± 0.92	3.80 ± 1.16	<u>2.14 ± 0.54</u>

Table S3: **Mean absolute error in accuracy estimation on binary tasks.** .

Dataset	n_ℓ	n_u	Labeled	Pseudo-Labeling (LR)	Dawid-Skene	Bayesian-Calibration	SSME-KDE-M	SSME-KDE (Ours)
Critical Outcome	20	1000	11.01 ± 4.04	6.94 ± 2.30	<u>2.61 ± 0.33</u>	3.48 ± 2.76	3.17 ± 1.10	**1.16 ± 0.48**
	50	1000	6.22 ± 2.23	5.38 ± 1.40	<u>2.37 ± 0.32</u>	2.56 ± 1.57	3.01 ± 0.94	**1.13 ± 0.47**
	100	1000	4.20 ± 1.38	3.63 ± 0.77	2.25 ± 0.37	<u>1.69 ± 0.91</u>	2.81 ± 0.81	**1.15 ± 0.38**
ED Revisit	20	1000	8.37 ± 3.14	4.16 ± 2.96	3.25 ± 2.45	3.57 ± 2.78	<u>1.88 ± 0.86</u>	**0.76 ± 0.16**
	50	1000	4.82 ± 1.73	2.29 ± 1.69	2.29 ± 1.68	2.04 ± 1.66	<u>1.83 ± 0.67</u>	**0.73 ± 0.18**
	100	1000	3.29 ± 0.88	1.36 ± 0.78	1.34 ± 0.76	<u>1.16 ± 0.73</u>	1.51 ± 0.63	**0.73 ± 0.21**
Hospital Admission	20	1000	21.76 ± 4.18	8.10 ± 4.61	17.31 ± 0.42	<u>5.12 ± 3.94</u>	5.54 ± 1.32	**1.97 ± 0.47**
	50	1000	12.74 ± 2.25	5.02 ± 2.45	16.60 ± 0.43	<u>3.49 ± 2.05</u>	5.20 ± 1.19	**2.06 ± 0.67**
	100	1000	8.56 ± 1.39	3.91 ± 1.76	15.62 ± 0.44	<u>3.23 ± 1.68</u>	5.32 ± 1.32	**1.70 ± 0.54**
SARS-CoV Inhibition	20	1000	7.44 ± 3.44	5.96 ± 3.13	4.35 ± 0.53	**2.24 ± 1.19**	<u>2.57 ± 0.64</u>	3.38 ± 0.47
	50	1000	3.66 ± 1.80	3.06 ± 1.28	4.08 ± 0.57	**1.73 ± 0.93**	<u>2.27 ± 0.72</u>	3.41 ± 0.41
	100	1000	2.18 ± 1.14	2.36 ± 0.78	3.67 ± 0.59	**1.35 ± 0.78**	<u>1.79 ± 0.69</u>	3.44 ± 0.47
Toxicity Detection	20	1000	5.85 ± 2.89	5.09 ± 2.87	<u>4.40 ± 0.33</u>	4.69 ± 1.21	5.67 ± 0.68	**2.35 ± 0.46**
	50	1000	3.99 ± 2.28	<u>3.04 ± 1.66</u>	4.20 ± 0.26	3.97 ± 1.21	4.57 ± 0.93	**2.26 ± 0.44**
	100	1000	2.37 ± 1.35	**1.91 ± 0.99**	4.10 ± 0.32	3.30 ± 0.91	3.43 ± 1.05	<u>2.19 ± 0.53</u>

Table S4: **Mean absolute error in ECE estimation on binary tasks.**

Dataset	n_ℓ	n_u	Labeled	Pseudo-Labeling (LR)	Dawid-Skene	Bayesian-Calibration	SSME-KDE-M	SSME-KDE (Ours)
Critical Outcome	20	1000	10.09 ± 4.84	31.73 ± 3.95	9.39 ± 1.25	<u>2.84 ± 0.91</u>	4.72 ± 2.27	**2.52 ± 1.24**
	50	1000	7.50 ± 4.62	27.33 ± 5.51	8.49 ± 1.46	<u>3.17 ± 1.17</u>	5.61 ± 4.61	**2.39 ± 1.74**
	100	1000	5.65 ± 3.44	20.43 ± 4.38	7.97 ± 1.08	**2.70 ± 0.94**	3.82 ± 1.72	<u>2.83 ± 2.89</u>
ED Revisit	20	1000	18.48 ± 6.68	<u>7.48 ± 0.72</u>	8.27 ± 3.80	7.65 ± 0.55	11.89 ± 4.66	**5.92 ± 3.14**
	50	1000	17.37 ± 7.13	7.48 ± 0.95	7.62 ± 0.99	<u>7.30 ± 0.76</u>	11.99 ± 4.36	**5.09 ± 2.56**
	100	1000	14.13 ± 6.03	<u>7.06 ± 1.46</u>	7.09 ± 1.52	7.47 ± 1.17	11.28 ± 5.73	**5.08 ± 2.77**
Hospital Admission	20	1000	6.97 ± 4.64	8.94 ± 5.97	16.70 ± 0.31	<u>2.67 ± 1.15</u>	3.63 ± 1.95	**2.51 ± 1.38**
	50	1000	5.08 ± 3.49	5.59 ± 4.31	16.18 ± 0.31	<u>2.62 ± 1.65</u>	3.18 ± 1.95	**2.51 ± 1.20**
	100	1000	3.57 ± 2.58	3.66 ± 2.68	15.32 ± 0.39	<u>2.55 ± 1.34</u>	3.17 ± 1.60	**2.02 ± 1.20**
SARS-CoV Inhibition	20	1000	9.61 ± 9.22	30.92 ± 4.35	7.50 ± 1.05	**3.07 ± 1.00**	5.42 ± 2.63	<u>3.48 ± 1.58</u>
	50	1000	5.84 ± 3.64	22.71 ± 4.29	7.06 ± 1.04	<u>3.62 ± 0.97</u>	5.02 ± 1.86	**3.41 ± 1.68**
	100	1000	3.97 ± 1.97	16.33 ± 3.27	6.04 ± 1.17	<u>3.53 ± 1.35</u>	4.21 ± 1.90	**3.46 ± 1.63**
Toxicity Detection	20	1000	6.71 ± 3.57	17.33 ± 7.52	6.20 ± 0.41	<u>5.22 ± 0.59</u>	6.05 ± 1.02	**3.34 ± 0.82**
	50	1000	<u>4.76 ± 3.29</u>	11.79 ± 6.41	5.97 ± 0.33	4.76 ± 0.74	4.86 ± 1.03	**3.15 ± 0.66**
	100	1000	<u>3.82 ± 2.17</u>	7.54 ± 3.73	5.84 ± 0.44	4.25 ± 0.96	4.15 ± 1.20	**3.09 ± 0.81**

Table S5: **Mean absolute error in AUC estimation on binary tasks.**

Dataset	n_ℓ	n_u	Labeled	Pseudo-Labeling (LR)	Dawid-Skene	Bayesian-Calibration	SSME-KDE-M	SSME-KDE (Ours)
Critical Outcome	20	1000	32.86 ± 18.26	22.98 ± 6.69	39.02 ± 4.26	<u>9.29 ± 6.01</u>	11.48 ± 5.46	**6.11 ± 2.63**
	50	1000	22.81 ± 13.16	20.48 ± 8.04	35.64 ± 5.80	<u>9.34 ± 5.17</u>	11.98 ± 5.17	**6.17 ± 3.47**
	100	1000	15.71 ± 8.81	14.45 ± 7.30	33.31 ± 4.33	<u>8.96 ± 5.07</u>	11.30 ± 6.01	**5.77 ± 2.80**
ED Revisit	20	1000	19.18 ± 13.27	5.14 ± 3.20	5.12 ± 4.68	9.14 ± 3.74	<u>5.07 ± 2.89</u>	**1.67 ± 1.06**
	50	1000	8.85 ± 8.14	<u>2.72 ± 2.22</u>	3.04 ± 2.80	6.03 ± 2.95	3.79 ± 2.33	**1.81 ± 1.08**
	100	1000	6.34 ± 5.57	**1.57 ± 1.19**	<u>1.74 ± 1.24</u>	4.23 ± 1.97	3.92 ± 2.23	1.82 ± 1.13
Hospital Admission	20	1000	9.43 ± 5.85	10.89 ± 9.33	21.15 ± 0.52	5.26 ± 3.84	<u>4.36 ± 1.60</u>	**3.47 ± 2.04**
	50	1000	7.46 ± 4.74	7.91 ± 5.89	20.34 ± 0.59	4.43 ± 2.68	<u>3.70 ± 2.26</u>	**3.64 ± 2.16**
	100	1000	5.51 ± 3.48	4.12 ± 3.66	19.17 ± 0.68	<u>3.49 ± 2.17</u>	4.00 ± 2.19	**2.80 ± 1.84**
SARS-CoV Inhibition	20	1000	22.27 ± 10.94	37.41 ± 8.82	16.60 ± 3.91	**7.54 ± 2.74**	<u>13.81 ± 5.52</u>	20.51 ± 6.12
	50	1000	15.02 ± 8.77	30.29 ± 9.40	15.01 ± 3.85	**8.40 ± 3.45**	<u>12.82 ± 3.63</u>	21.06 ± 5.46
	100	1000	11.53 ± 5.64	20.34 ± 6.46	12.61 ± 3.78	**8.27 ± 3.19**	<u>11.01 ± 5.02</u>	20.67 ± 5.92
Toxicity Detection	20	1000	<u>19.34 ± 8.45</u>	25.13 ± 12.71	26.34 ± 1.31	19.94 ± 4.70	23.38 ± 2.64	**10.89 ± 3.14**
	50	1000	<u>13.78 ± 6.52</u>	20.15 ± 12.57	25.24 ± 1.31	16.84 ± 5.68	18.90 ± 3.88	**9.91 ± 3.06**
	100	1000	<u>10.69 ± 6.16</u>	14.06 ± 7.21	24.51 ± 1.68	14.15 ± 5.35	14.59 ± 4.54	**9.88 ± 3.51**

Table S6: **Mean absolute error in AUPRC estimation on binary tasks.**

Dataset	n_ℓ	n_u	Labeled	Pseudo-Labeling	Dawid-Skene	AutoEval	SSME-KDE	SSME-NF
AG News	20	1000	5.79 ± 3.04	5.72 ± 4.16	8.31 ± 0.54	5.61 ± 2.77	**2.77 ± 0.96**	5.56 ± 0.75
	50	1000	4.09 ± 1.92	2.97 ± 2.00	8.06 ± 0.68	3.68 ± 1.48	**2.72 ± 1.09**	5.64 ± 1.03
	100	1000	2.93 ± 1.52	**2.36 ± 1.48**	7.66 ± 0.69	2.70 ± 1.29	2.50 ± 1.09	5.32 ± 1.05
ImagenetBG	20	1000	6.62 ± 2.74	33.45 ± 2.96	5.78 ± 0.71	6.55 ± 2.62	8.76 ± 1.00	**2.65 ± 0.67**
	50	1000	3.98 ± 1.63	17.88 ± 2.78	5.69 ± 0.73	3.87 ± 1.56	8.18 ± 0.90	**2.66 ± 0.81**
	100	1000	2.97 ± 1.38	9.37 ± 1.53	5.34 ± 0.63	2.73 ± 1.13	8.02 ± 0.90	**2.10 ± 0.68**
MultiNLI	20	1000	7.46 ± 3.88	7.95 ± 4.55	11.73 ± 0.55	7.20 ± 3.76	**1.98 ± 0.88**	3.08 ± 0.65
	50	1000	4.42 ± 1.99	3.08 ± 2.25	11.41 ± 0.52	4.17 ± 1.96	**1.90 ± 0.76**	2.79 ± 0.81
	100	1000	3.27 ± 1.65	2.47 ± 1.86	10.72 ± 0.54	3.17 ± 1.59	**2.02 ± 0.82**	2.52 ± 0.77

Table S7: **Mean absolute error in accuracy estimation on multiclass tasks.**

Dataset	n_ℓ	n_u	Labeled	Pseudo-Labeling	Dawid-Skene	SSME-KDE	SSME-NF
AG News	20	1000	7.04 ± 2.22	4.48 ± 3.23	5.60 ± 0.28	**2.24 ± 0.51**	3.72 ± 0.50
	50	1000	4.85 ± 1.54	2.28 ± 1.36	5.37 ± 0.34	**2.24 ± 0.59**	3.81 ± 0.46
	100	1000	3.24 ± 1.15	**1.89 ± 0.96**	5.02 ± 0.40	2.15 ± 0.55	3.53 ± 0.60
ImagenetBG	20	1000	7.10 ± 2.79	29.64 ± 2.84	4.76 ± 0.56	6.73 ± 0.54	**2.49 ± 0.60**
	50	1000	4.00 ± 1.85	14.18 ± 2.51	4.68 ± 0.48	6.42 ± 0.57	**2.49 ± 0.72**
	100	1000	2.75 ± 1.13	6.68 ± 1.04	4.54 ± 0.52	6.30 ± 0.62	**1.96 ± 0.60**
MultiNLI	20	1000	11.57 ± 4.06	7.84 ± 4.12	2.95 ± 0.29	2.06 ± 0.88	**1.75 ± 0.62**
	50	1000	6.14 ± 2.42	3.10 ± 2.24	2.92 ± 0.37	2.06 ± 0.72	**1.63 ± 0.59**
	100	1000	4.52 ± 1.83	2.37 ± 1.66	3.18 ± 0.30	2.19 ± 0.76	**1.45 ± 0.57**

Table S8: **Mean absolute error in ECE estimation on multiclass tasks.**

Using the semisynthetic setting described in Section 6.4, we artificially increase the expected calibration error of each classifier using a generalized logistic function parameterized by a. Specifically, we transform classifier score s to be $\frac{s^a}{s^a+(1-s)^a}$, effectively increasing overconfidence for higher s and increasing underconfidence for lower s. As in the semisynthetic experiments, we generate 500 semisynthetic classifier sets, where each classifier in a set is trained on 100 examples distinct from the training data for other classifiers in the set (results are robust to this choice of training dataset size). Each set contains three classifiers.

Figure S3 reports our results. As the average calibration among classifiers in a set varies, SSME consistently improves over the use of an ensemble. This aligns with our intuition, and indicates the value of using labeled data in conjunction with unlabeled data. Interestingly, miscalibration has little effect on the ensemble when estimating AUPRC; here, SSME and ensembling perform similarly.

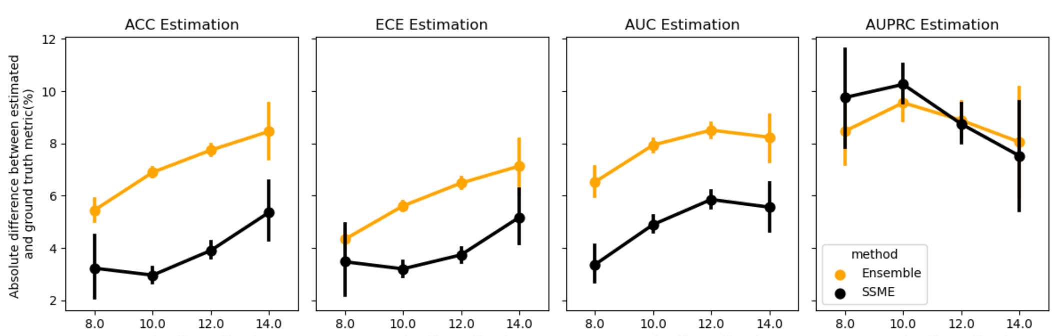

Figure S3: **A comparison of SSME to ensembling on a miscalibrated classifier set.** SSME consistently produces more accurate performance estimates compared to ensembling the classifiers across differently calibrated classifier sets (x-axis).

E Method Details

E.1 Metric Estimation

Given a vector $p \in \Delta^{K-1}$ over K classes, let $\mathbf{s} = \text{ALR}(\mathbf{p}) = \left[\log \frac{\mathbf{p}_1}{\mathbf{p}_K}, \log \frac{\mathbf{p}_2}{\mathbf{p}_K}, \cdots, \log \frac{\mathbf{p}_{K-1}}{\mathbf{p}_K}\right] \in \mathbb{R}^{K-1}$. To invert, $\mathbf{p}_i = \frac{e^{\mathbf{s}_i}}{1+\sum_{k=1}^{K-1} e^{\mathbf{s}_k}}$ for $i < K$ and $\mathbf{p}_K = \frac{1}{1+\sum_{k=1}^{K-1} e^{\mathbf{s}_k}}$. The ALR transform maps unit-sum data into real space, where it is easier to fit mixture models. The inverse allows us to map samples from the mixture model in real space back to the simplex Δ^{K-1}. For details, see Pawlowsky-Glahn & Buccianti (2011).

E.2 Metric Estimation

SSME is able to estimate any metric that is a function of the classifier probabilities p and label y. We approximate the joint distribution $P(y, \mathbf{p})$ with a mixture model model $P_\theta(y, \mathbf{s})$, where $\mathbf{s}$ refers to the ALR-transformed classifier probabilities (i.e. "classifier scores")[2]. We refer to $P(y, \mathbf{p})$ for ease of notation in this section; it is equivalent, through invertible mapping, to $P(y, \mathbf{s})$.

We denote our approximation for $P(\mathbf{p}, y)$ as $P_\theta(\mathbf{p}, y)$. We provide a few concrete examples of how one can use SSME to measure performance metrics, given $P_\theta(\mathbf{p}, y)$ and a set of unlabeled probabilistic predictions $\{\mathbf{p}^{(i)}\}_{i=1}^{n_u}$ and labeled probabilistic predictions $\{\mathbf{p}^i, y^{(i)}\}_{i=1}^{n_\ell}$. Notationally, $\mathbf{p}_j^i$ refers to the jth model's probabilistic prediction of the ith unlabeled example.

Accuracy measures the alignment between a model's (discrete) predictions and the true label y. To discretize predictions, practitioners typically take the argmax of $\mathbf{p}^{(i)}$. Using the binary case an illustrative example, the accuracy of the jth model can be written as:

$$\text{Accuracy}_j = \mathbb{E}_{\mathbf{p}}\left[\mathbf{1}\left[y = \mathbf{1}(\mathbf{p} > t)\right]\right]$$

where $\mathbf{1}$ is an indicator function and t is a chosen threshold, typically 0.5. In our setting, we approximate this as:

$$\text{Accuracy}_j \approx \frac{1}{n_u + n_\ell} \sum_{i=1}^{n_u+n_\ell} \mathbf{1}\left[y^{(i)} = \mathbf{1}(\mathbf{p}^{(i)} > t)\right]$$

For labeled examples, we use the true label $y^{(i)}$. For unlabeled examples, we draw $y^{(i)} \sim P_\theta(y|\mathbf{p}^{(i)})$. We then compute accuracy using these labels $y^{(i)}$ and predictions $\mathbf{p}^{(i)}$. To ensure our estimation procedure is robust to sampling noise, we average our estimated accuracy over 500 separate sampled labels for each example in the unlabeled dataset.

Alternatively, we could directly use $P_\theta(y|\mathbf{p})$ to estimate accuracy. That is, for each point $\mathbf{p}^{(i)}$ we directly compute an expectation for the label, and sum this over the entire dataset.

Using the binary case as an example

$$\text{Accuracy}_j \approx \frac{1}{n_u + n_\ell} \sum_{i=1}^{n_u+n_\ell} \mathbb{E}\left[\mathbf{1}\left[y^{(i)} = \mathbf{1}(\mathbf{p}_j^{(i)} > t)\right] | \mathbf{p}^{(i)}\right]$$

In other words, we compute the expectation that the true label agrees with the predicted label for each point . This expectation is $\mathbf{p}^{(i)}$. This expectation is computed over $P_\theta(y|\mathbf{p})$ One can interpret $P_\theta(y|\mathbf{p})$ as a "recalibration" step: given a set of classifier guesses $\mathbf{p}$, what is the true distribution of y?

In our experiments, we use the first of these two approaches, i.e. we sample the true label from the estimated distribution.

Expected Calibration Error (ECE) measures the alignment between a model's predicted probabilities $\mathbf{p}_j$ and the ground truth labels y. In particular, ECE compares the model's reported confidence

[2]Recall that ALR is a bijection, so we use the inverse mapping $\text{ALR}^{-1} : \mathbb{R}^{K-1} \to \Delta^{K-1}$ to transform our mixture distribution in real space back to probability space.

to the true class likelihoods, averaged over the dataset. We write out our ECE estimation procedure for the binary case, and it extends readily to definitions of calibration in multiclass settings (Gupta & Ramdas, 2022). Binary ECE can be written as:

$$\text{ECE}_j = \mathbb{E}_{\mathbf{p}_j}\left[\left|P(\hat{Y}=1|\hat{p}=\mathbf{p}_j)-\mathbf{p}_j\right|\right]$$

Then, to approximate the ECE with the datasets $\{\mathbf{p}^i\}_{i=1}^{n_u}$ and $\{\mathbf{p}^i, y^{(i)}\}_{i=1}^{n_\ell}$, one can sample $y^{(i)} \sim P_\theta(y|\mathbf{p}^{(i)})$ for each unlabeled sample i and then use the standard histogram binning procedure (Guo et al., 2017) using both the true labels for the labeled dataset and the sampled labels for the unlabeled dataset. In this approach, we treat the sampled labels $y^{(i)}$ as true labels for unlabeled examples. To ensure our procedure is robust against sampling noise, we draw samples of $y^{(i)}$ repeatedly for a fixed number of draws (500). We then compute ECE separately for each of these 500 draws and average ECE across all draws.

Alternatively, one could also *directly* use $P_\theta(y|\mathbf{p})$ to estimate ECE. In particular, we can write:

$$\text{ECE}_j \approx \frac{1}{n_u+n_\ell}\sum_{i=1}^{n_u+n_\ell}\left|P_\theta\left(y=1|\mathbf{p}_j^{(i)}\right)-\mathbf{p}_j^{(i)}\right|$$

In this approach, we don't sample the labels y for unlabeled examples but instead directly use $P_\theta(y|\mathbf{p})$, which provides us (an estimate of) the true distribution of y. Instead, we directly use our estimate for the conditional label distribution $P_\theta\left(y=1|\mathbf{p}_j^{(i)}\right)$. In our experiments, we use the first approach described, i.e. sampling $y^{(i)}$ for unlabeled examples and then using the standard binning and averaging procedure.

AUROC and AUPRC can be estimated with a similar procedure as above. In particular, we sample a label $y^{(i)} \sim P_\theta\left(y=1|\mathbf{p}^{(i)}\right)$ from the conditional label distribution and compare these sampled labels to the classifier probabilities.