Table 3: Performance of SMS-DKL (optimized over the baseline LSTM model) and Single-model benchmarks (with various modifications to the baseline LSTM model). Results are computed on (held-out) test data; in line with the original paper Oh et al. (2019), we include the area under the receiver operating characteristic (AUROC) and the area under the precision-recall curve (AUPRC), each shown with 95% confidence intervals. Results for SMS-DKL are computed at after 500 BO iterations; note that these numbers (computed on test data) are not exactly identical to those in Table 2 (computed on validation data).

Target	IHM		Shock		ARF	
Metric	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC
LSTM	0.80 [0.78, 0.83]	0.39 [0.33, 0.43]	0.59 [0.49, 0.69]	0.09 [0.05, 0.16]	0.47 [0.35, 0.58]	0.04 [0.02, 0.07]
LSTM+t	0.81 [0.79, 0.83]	0.41 [0.36, 0.47]	0.62 [0.53, 0.70]	0.08 [0.05, 0.15]	0.42 [0.30, 0.54]	0.04 [0.02, 0.07]
LSTM+TE	0.82 [0.80, 0.85]	0.43 [0.38, 0.48]	0.60 [0.50, 0.69]	0.10 [0.06, 0.20]	0.48 [0.35, 0.61]	0.05 [0.03, 0.10]
HyperLSTM	0.82 [0.80, 0.84]	0.42 [0.37, 0.47]	0.63 [0.54, 0.72]	0.08 [0.05, 0.12]	0.57 [0.44, 0.68]	0.06 [0.03, 0.10]
shiftLSTM	0.81 [0.79, 0.84]	0.43 [0.37, 0.48]	0.61 [0.52, 0.70]	0.09 [0.05, 0.16]	0.61 [0.49, 0.70]	0.10 [0.03, 0.21]
mixLSTM	0.83 [0.81, 0.85]	0.45 [0.40, 0.50]	**0.67 [0.58, 0.76]**	0.10 [0.06, 0.16]	**0.72 [0.62, 0.80]**	**0.15 [0.06, 0.27]**
SMS-DKL	**0.84 [0.82, 0.87]**	**0.46 [0.40, 0.53]**	0.65 [0.59, 0.72]	**0.12 [0.07, 0.17]**	0.66 [0.61, 0.72]	0.11 [0.07, 0.15]

A Implementation Details

Implementation. The benchmarks GP and ParEGO (and their stepwise variants) are implemented using the `GPyOpt` library González (2016). All models use a Matérn-5/2 covariance kernel and automatic relevance determination hyperparameters, optimized by empirical Bayes Williams and Rasmussen (2006). For these algorithms as well as SMS-DKL, we use the expected improvement (EI) Jones et al. (1998); Mockus et al. (1978) as the acquisition function. The DKL network consists of three components as shown in Figure 3. The RNN component is implemented as an LSTM network with 50 hidden units per cell, using tanh as hidden recurrent activation and sigmoid as output activation. The DeepSets component is implemented as a four-layer feedforward network with 32 hidden units per layer and ReLU as activation; we average all the samples in the dataset after the transformation of the second hidden layer. See Zaheer et al. (2017) for additional details on DeepSets and permutation invariance. In our experiments, we set the learned embedding of the filtration at each time step to be of size 1. As for the MLP component in the DKL network, we use a three-layer feedforward network with 32 hidden units per layer and tanh as activation. In training, we set the maximum number of training iterations $M = 500$. The optimizer we use is Adam Kingma and Ba (2014). The benchmark PESMO is implemented using the source code provided by Hernández-Lobato et al. (2016).[2] The size of the Pareto set sample in PESMO is 50, which is used and claimed to be suitable for several hundreds of acquisitions in the original PESMO paper. For all BO algorithms, we set the maximum number of function evaluations $N = 500$ and use the same initial sample for all the algorithms.

[2]https://github.com/HIPS/Spearmint/tree/PESM

B Hyperparameter Space

In Section 5, we evaluate the proposed SMS-DKL algorithm and all benchmark algorithms in optimizing the hyperparameters of RNN models on 7 sequence prediction tasks. The RNN architecture used in the experiment is a standard LSTM network with tanh as hidden recurrent activation and a hidden ReLU output layer for making predictions. The full eight-dimensional hyperparameter space $\mathbb{X}$ of RNN models is as follows,

- Number of output units: int, [10, 200]
- RNN state size: int, [10, 300]
- Training epochs: int, [32, 200]
- Batch size: int, [32, 100]
- Dropout rate: float, [0.1, 0.9]
- Recurrent dropout rate: float, [0.1, 0.9]
- Logarithm of learning rate: int, [-8, -3]
- Logarithm of weight decay: int, [-20, 1]

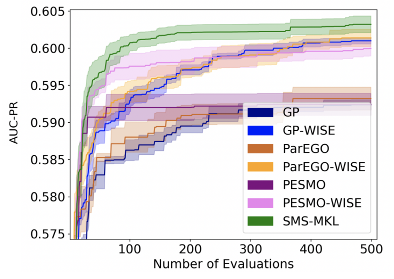
C Generalization Performance

In the main manuscript, we primarily focused on comparing SMS-DKL to other algorithms in the context of BO—that is, with respect to the performance of selected hyperparameters on validation data. Now in practice, a reasonable question is whether these stepwise hyperparameters—selected by SMS-DKL on the validation set—can generalize well to (held-out) test data. In particular, a variety of sophisticated *single-model* techniques have been proposed to address the problem of temporal distribution shift by modifying the baseline LSTM model. These explicitly include time as a parameter (LSTM+t), incorporate temporal encodings (LSTM+TE), model abrupt transitions (shiftLSTM), mix weights over time (mixLSTM), as well as learning hypernetworks to modify the weights of the LSTM model (HyperLSTM); we refer to Oh et al. (2019) for a more detailed treatment. Here, we include a head-to-head comparison of such single-model techniques with SMS-DKL—applied over the baseline LSTM alone. Table 3 shows results for all aforementioned single-model techniques designed to accommodate time-varying relationships, as well as the result of SMS-DKL (applied to the baseline LSTM model). Results for the single-model techniques are reprinted from Oh et al. (2019), and include all of the same prediction tasks on the MIMIC dataset. Results for SMS-DKL are obtained via the same training and testing splits; we execute the same data processing procedure using the source code accompanying the original paper.[3] On the one hand, the various modifications to the LSTM model clearly improve performance over the baseline LSTM model. On the other hand, we observe that extremely competitive performance is also achieved by the simple application of SMS-DKL in optimizing the baseline LSTM model: SMS-DKL exhibits either the best or second-best test-set performance—purely by optimizing the (unmodified) LSTM baseline. SMS-DKL lays the foundation for AutoML to offer a general and convenient framework of developing powerful sequence prediction models while keeping the human out of the loop.

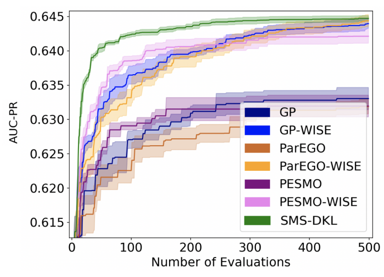
D Convergence Plots

In this section, we provide the convergence plots of all BO algorithms over all 500 evaluations, for each of the prediction tasks corresponding to Table 2 in Section 5.

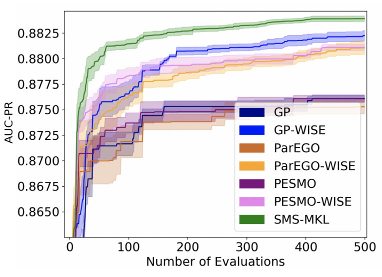
[3]https://gitlab.eecs.umich.edu/mld3



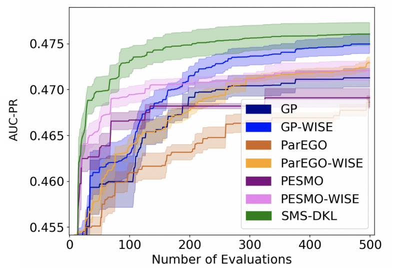
(a) UKCF - 1YM



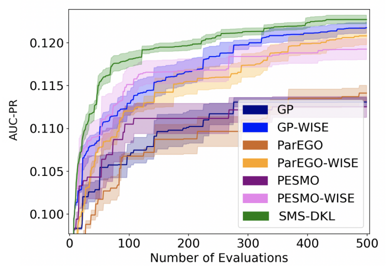
(b) UKCF - ABPA



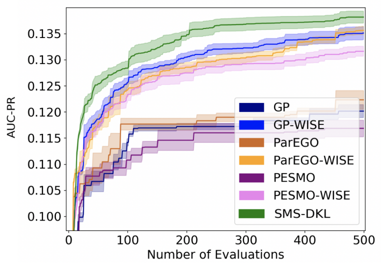
(c) UKCF - E.coli



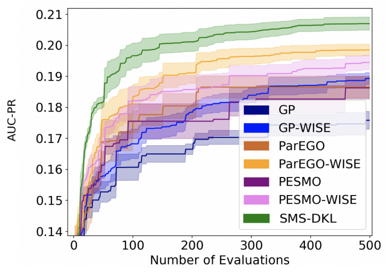
(d) MIMIC - IHM



(e) MIMIC - Shock



(f) MIMIC - ARF



(g) WARDS - ICU

Figure 5: Convergence plots of all BO algorithms over all 500 evaluations, for each of the seven prediction tasks in Table 2.

E Table and Pseudo code

Dataset	Target	Autocorr.	% Postive (start)	(end)
UKCF	1YM	0.592	12.2	23.4
	ABPA	0.564	27.6	12.7
	E. coli	0.564	52.4	28.5
MIMIC	IHM	0.849	13.2	13.2
	Shock	0.867	8.9	8.9
	ARF	0.875	7.2	7.2
WARDS	ICU	0.976	1.7	3.2

Table 4: Feature autocorrelations and % positive labels. The slight inter-task variation in feature autocorrelations within a dataset are due to label missingness and censoring.

Algorithm 1 SMS-DKL

```
**Hyperparameters:** Max BO iterations $N$, max training iterations $M$, and acquisition function $a_f$
**Input:** Sequence dataset $\mathcal{D}$
Initialize $\mathcal{A}_t$, $t \in \{1, ..., T\}$ with random samples
for $n = 1$ to $N$ do
  for $m = 1$ to $M$ do
    Update $\mathbf{\Theta}_t$, $t \in \{1, ..., T\}$ jointly
        by optimizing (5)
  end for
  Update $a_{f,t}(\mathbf{x}_t|\mathcal{A}_t)$, $t \in \{1, ..., T\}$ using (7)
  Solve $\mathbf{x}_t^* = \arg\max_{\mathbf{x}_t \in \mathbb{X}} a_{f,t}(\mathbf{x}_t|\mathcal{A}_t)$, $t \in \{1, ..., T\}$
  Sample $\mathbf{x}^*$ from $\{\mathbf{x}_t^* : t \in \{1, ..., T\}\}$
      via policy in (9)
  $\mathcal{A}_t \leftarrow \mathcal{A}_t \cup (\mathbf{x}^*, y_t^*)$, $t \in \{1, ..., T\}$
end for
**Output:** For each $t \in \{1, ..., T\}$, the tuple $(\mathbf{x}_t, y_t)$ with the best value of $y_t$ in $\mathcal{A}_t$
```