Incorporating downstream, task-specific information in forecasting models

Hussain Kazmi, and Maria Paskevich

January 31, 2023

Abstract

Forecasts underpin most modern planning activities. However, an age-old question persists in their creation: should forecast models prioritize accuracy or downstream utility? This question has been debated for well over a century, with diverging opinions in the community, spawning a plethora of evaluation metrics. Probabilistic or interval forecasts can enable planners to make risk-informed decisions, but often do so at elevated computational (and conceptual) costs. In the meantime, Predict+Optimize workflows have witnessed renewed attention in recent years. These methods integrate downstream task information in the prediction loop to adapt the forecast model in a principled manner. Often, this is done through direct loss function augmentation, i.e. while training the forecast model. Even though this can greatly improve downstream utility, it comes at much greater computational cost since an expensive optimization problem (i.e. the downstream task) has to be solved during each iteration of the learning algorithm (e.g. a boosted tree or neural network). As modern algorithms typically require hundreds or thousands of iterations to properly tune their parameters, this quickly becomes intractable. Hyperparameter search to identify the best models further exacerbates the situation. Mechanistically, direct loss function augmentation also requires algorithm and library specific tweaks.

In this report, we propose and utilize a Bayesian Optimization based framework which can perform the Predict+Optimize task in a computationally efficient and model-agnostic manner. By integrating the downstream function only in the hyperparameter search routine, we show it is possible to (1) significantly improve model accuracy and downstream performance, irrespective of the learning algorithm, and (2) reduce the computational complexity of the Predict+Optimize operation by several orders of magnitude. We validate these findings using two different case studies in energy and retail supply chain optimization. Our results show that the proposed methodology works well for both the case when accuracy is well correlated with accuracy, as well as when it is not.

Contents

1 Introduction

In a seminal paper published in 1950, Glenn Brier wrote 'the verification of weather forecasts has been a controversial subject for more than a half century' [1]. In the paper, Brier posited that a core challenge with forecast verification is the frequent disconnect between accuracy and utility: i.e. the most accurate forecast may not necessarily be the most useful one. This realization can lead to the forecaster hedging their prediction by forecasting something other than what they think will actually happen.

Is such an adaptation fair, or even useful, especially if done in an ad-hoc manner? On the one hand, a forecaster should make the most accurate prediction they are capable of. On the other, a forecast is not an island. Instead, it typically finds downstream uses, which determine its utility. In the decades since Brier's musings, the debate has only grown louder, with the field vacillating

between scores of different forecast metrics which have been proposed both to train and verify forecast models [2]. Two developments in this regard deserve a mention:

1.1 Judgment-based modifications

Judgment-based modifications to forecasts typically rely on a human domain expert adjusting forecasts, often produced using software [3]. This is presumably done to serve as either a sanity check or a utility check. The former is meant to ensure that the forecast follows domain knowledge, while the latter targets an improvement in the forecast's downstream utility by baking in some measure of 'risk', as informed by expert knowledge. However, these adjustments are often driven by biases, hopes, and fears, rather than rigorous practice. Consequently, they often end up worsening performance on one or both of forecast accuracy and downstream utility. In fact, a large-scale survey of over a hundred practitioners led Goodwin and Fildes to conclude that most firms would greatly benefit from 'requiring managers to justify their adjustments in writing, and assessing the results of judgmental interventions' [4]. Fifteen years later, it is doubtful whether this advice has been adopted across the board.

1.2 Probabilistic or interval forecasts

Probabilistic or interval forecasts eschew predicting a single value at every future time point to produce the entire distribution instead [5]. This provides a clearer view of the future, especially when the mean value, as is often reported in point forecasts, does not make logical sense, e.g. for intermittent demand or binary outcomes. At the same time, producing interval forecasts is more complex (both in the creation and evaluation phase), and introduces additional downstream complexity in the decision-making process. Furthermore, producing well-calibrated interval forecasts, i.e. correctly predicting the entire distribution, especially when only sparse data is available, is extremely challenging. This is further exacerbated by the concern that improbable events occur only very rarely by definition, so a hedge might need to be built even into interval forecasts for risk-constrained downstream optimization.

1.3 Brief literature review and contributions

These two developments notwithstanding, the core question remains the same: should forecasters concern themselves with downstream utility when producing forecasts, and if so how can this be done in a principled manner that does not lead to loss of accuracy or utility (or both)? Very recently, this topic has seen renewed interest from the research literature. A notable example of this was the 3rd IEEE Technical Challenge on Predict+Optimise, i,e. joint energy forecasting and scheduling activities [6]. This has led to an increased interest in understanding the effects of under- and over-forecasting [7], as well as in designing custom loss functions by task-specific information augmentation while training the forecasters [8]. Where the model learns by backpropagating the prediction error, this augmentation can leverage advances in the techniques used to create machine learning models such as neural networks.

While the idea of backpropagating downstream utility might seem appealing, it poses significant computational challenges. On the one hand, downstream tasks are often ill-posed, i.e. they do not have an analytic or closed-form solution. This means that there is no guarantee that these tasks will be amenable to convex optimization techniques, which still have efficient solvers. Furthermore, the downstream utility function is often vaguely defined or poorly observed, which can lead to further challenges in training models in this way. Nevertheless, where this information is available and it

can be used efficiently, it is plausible that using it while training or selecting forecast models will bring an improvement in the forecaster's utility.

In light of these observations, we propose an end-to-end model-agnostic Bayesian Optimization framework to create computationally efficient task-informed forecasters (i.e. models for the entire Predict+Optimize loop). Bayesian Optimization (BO) refers to a family of derivative-free algorithms that utilize an intelligent sampling scheme to optimize a given function, based on the value of previously collected samples [9]. In doing so, BO is much more computationally efficient than naive methods, while remaining model agnostic. This has the added benefit of determining which features are the most important for both accuracy and downstream utility. To test this idea, we ran a series of experiments in two different domains: energy and retail, which are documented in this report.

1.4 Organization of the report

The remainder of this report is organized as follows: Section 2 presents a brief overview of the methodology we follow in the remainder of the paper. Section 3 describes the two case studies where we apply the proposed methodology. Section 4 includes practical implementation details. Section 5 presents results from the two case studies, while Section 6 includes a discussion on the key insights from the presented research, and concludes the report.

2 Methodology

In this section, we describe the methodology we followed in detail. We motivate the work using two different case studies, one in energy and another in the retail sector.

2.1 The classical machine learning based forecast pipeline

In recent years, forecasting literature has seen a rapid increase in the use of machine learning (ML) based techniques. Larger datasets, algorithmic advances, and more compute power make these algorithms competitive against classical models in many instances [10, 11]. From a practical perspective, such models typically minimize a loss function (often chosen as mean absolute error or mean squared error) on the one-step ahead forecast. This model can then be used recursively to produce arbitrary horizon forecasts. The loss function, formulated to minimize the mean absolute error, takes the following form:

$$\mathcal{L}_p = \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{1}$$

where $\mathcal{L}_p$, the sum of absolute prediction errors ($\sum |y_i - \hat{y}_i|$), is the loss function to be minimized given that y_i are the observations and $\hat{y}_i$ are the predictions in this case. Given some training data, this can be achieved in practice using a variety of optimizers, including gradient descent which is widely used when derivative information is available (e.g. when creating neural networks through backpropagation). In special cases, such as for ordinary least squares, the loss function (sum of squared errors) can be analytically minimized in one-shot. It is to be noted that despite their high bias, such recursive models are convenient abstractions to make arbitrary length forecasts. Their direct counterparts are not commonly used in practice due to the high costs of training and executing several models when making a single forecast [12].

When the training corpus is small, ML-based techniques often underperform simpler, classical counterparts such as Theta, ARIMA and even exponential smoothing due to overfitting. This is

typically addressed by introducing an additional regularization term in the loss function introduced above. The purpose of this term is to prevent the model from overfitting to random variations in the observation time series. The regularized loss function takes the following form:

$$\mathcal{L}_r = \underbrace{\sum_{i=1}^{N} |y_i - \hat{y}_i|}_{\text{Prediction term}} + \underbrace{\lambda_r \sum_{j=1}^{P} \beta_j}_{\text{Regularization term}} \tag{2}$$

where λ_r is the regularization strength (i.e. $\lambda_r = 0$ reduces to eq. 1), and β_j is the parameter set of the model (i.e. $\hat{y}_i = \beta x_i$, where x is an input feature vector). This is a special form of regularization known as L1 or Lasso regularization [**?**], which can be used to perform feature selection in addition to regularization (i.e. high values of λ_r will force many parameters of the model to take a value of zero). Other popular alternatives include Ridge or L2 regularization, as well as the Elastic Net formulation which combines both Ridge and Lasso loss terms with a weighting.

2.2 Co-optimization, or extended CASH

A key determinant for the accuracy of ML-based forecasters is in the choice of hyperparameters. Unlike model parameters which are fit using training data and a loss function, hyperparameters are factors which are set before the model training process begins. These hyperparameters include the choice and strength of regularization in eq. 2, but can also include the model architecture of neural networks, the depth and width of tree-based models, and the degree of interaction terms in simple regression models. While many techniques have been proposed in literature on the tuning of hyperparameters, the broader problem of Combined Algorithm Selection and Hyperparameter Optimization (CASH) has emerged as an interesting avenue of research in recent years [13, 14, 15]. This problem is typically posed as:

$$Find: A^*, \theta^* = \underset{A \in \mathcal{A}, \theta \in \Theta}{argmin} \mathbb{E}_{pt} \left[\mathcal{L}_{\mathcal{T}}(A_\theta(D_{train})) \right] \tag{3}$$

where $\mathcal{A}$ is the space of available algorithms (e.g. random forests, boosted trees, neural networks etc.), Θ is the space of all hyperparameters that are relevant to the available algorithms, $\mathcal{L}_{\mathcal{T}}$ is the loss function on the task $\mathcal{T}$ to be solved, and D_{train} is the training corpus. Note that the CASH problem reduces to the classical hyperparameter selection problem if we constrain the space of available algorithms $\mathcal{A}$ to contain a single model type. In this case, the model type can also be included as just another feature in the hyperparameters that need to be optimized. This latter formulation, while conceptually equivalent, requires conditional hyperparameters to be implemented.

In this report, we extend this CASH formulation, i.e. our objective is to determine the algorithm and hyperparameters that minimize not just the loss on the prediction task, but also the downstream decision-making task. This means the loss function now takes the form in eq. 4:

$$\mathcal{L} = \underbrace{\lambda_p \sum_{i=1}^{N} |y_i - \hat{y}_i|}_{\text{Prediction term}} + \underbrace{\lambda_r \sum_{j=1}^{P} \beta_j}_{\text{Regularization term}} + \underbrace{\lambda_t \sum_{i=1}^{N} C_i}_{\text{Task-related term}} \tag{4}$$

where the λ coefficients are now subscripted for the specific purpose they serve in the loss formulation (prediction, regularization and task-specific). The newly introduced C_i term denotes the downstream cost incurred at each time step due to misforecasts. More generally, we can now formulate the end-to-end Predict+optimize task as follows:

$$Find: f^* := \underset{f}{argmin} \left(\lambda_p \mathcal{L}_p\Big(y, f(x)\Big) + \lambda_r \mathcal{L}_r\Big(y, f(x)\Big) + \lambda_t \mathcal{L}_t\Big(y, f(x)\Big) \right) \tag{5}$$

In defining the training task this way, the mathematical formulation remains unchanged, but the complexity of finding the correct f^* grows considerably since it is now a nested optimization problem:

1. An initial forecast model is created, which is used to make a prediction;
2. This forecast is used to solve the optimal decision-making task for the entirety of the available dataset to attain a business-derived goal (e.g. to maximize profits or revenues, minimize costs etc.).

At the end of this loop, the forecast (model) is scored on not just the predictive accuracy but also downstream utility. Subsequently, model weights are adjusted in a way that reduces not just the prediction error but also downstream costs. Where the downstream task cannot be posed as a computationally efficient problem, this can quickly become tractable. However, a third loop is introduced in this formulation as well, in cases where model selection and hyperparameter optimization (i.e. the CASH formulation) needs to be performed as well. Given the fact that modern ML algorithms typically rely on several hyperparameters, this selection increases computational costs by several orders of magnitude. The second challenge with incorporating downstream task information in the training loop is that it can render algorithm-specific optimizations infeasible (this is especially valid for closed-form solutions).

2.3 The road to Bayesian Optimization

Given the high complexity (and, in many real-world cases, futility) of developing such a forecast model, we posit that a middle-ground should be preferable. In this approach, we hypothesise that the downstream costs should be integrated in the CASH step and not in the training phase of the learning algorithm. This should yield the desired benefits of creating task-aware forecasters while keeping the computational budget manageable. Bayesian Optimization (BO) has shown great promise in recent years for data-efficient black-box optimization tasks. However, before describing Bayesian Optimization techniques, we discuss other methods that could be used in such a CASH formulation, and highlight some of their key limitations.

2.3.1 Grid search

The simplest strategy, and one that has been around for many years despite its several well-documented shortcomings is grid search [16]. This search method defines a grid of hyperparameters and then runs through the training process with each of those hyperparameters individually. The set of hyperparameters that minimizes the loss is then chosen as the best model. It is important to point here that while grid search is used for selecting the best set of hyperparameters, it is not used to tune the parameters. That task is typically achieved using a more efficient algorithm (ranging from exact analytical solutions as in the case of ordinary least squares regression to gradient descent [17] and population based methods [18]).

2.3.2 Random search

Random search is a more data-efficient method for hyperparameter tuning, which has been popularized with the advent of modern neural networks featuring a very large number of hyperparameters to be tuned. This method varies from grid search in the sense that, instead of drawing samples on a rigid grid, it randomly samples hyperparameters in a pre-defined range. This makes its practical performance better than grid search, especially in case where some hyperparameters are irrelevant [19]. Another advantage of random search is that it is an anytime algorithm, i.e. the algorithm can be run for an arbitrary computational budget directly, and the entire grid does not need to be traversed. Nevertheless, the method remains completely memory-less: since it does not leverage the information from previous hyperparameter draws, it can be slow to converge to the optimal set of hyperparameters in practice.

2.3.3 Successive halving and variants

The successive halving technique can be used to address issues concerning the memory-less nature of grid and random search [20, 21]. This technique begins by generating a (random) population of candidate hyperparameter sets. After training separate models using this set of hyperparameters for a fixed budget (i.e. one model corresponding to one set of hyperparameters), it then prunes the least promising solutions and reallocates the budget of the pruned solutions to the retained solutions. Over the subsequent iterations, it repeats the same process over and over again, until only the best set of hyperparameters remains. This technique has the obvious benefit that it retains the most promising solutions, given the initial set of hyperparameters is sufficiently diverse. At the same time, by allocating more budget to promising solutions, it cuts down on unnecessary computation.

2.3.4 Bayesian Optimization (BO)

Some key limitations of the successive halving algorithm include (1) a reliance on the quality of the initial population, and (2) the risk of pruning good hyperparameters in case their convergence rate is slower than others. This latter is, for instance, the case when treating learning rate as a hyperparameter. Consequently, BO techniques have, in recent years, become increasingly popular to select the best hyperparameters for a learning algorithm [22, 23, 24, 25, 26]. These algorithms are also frequently extended to consider the entire CASH pipeline.

The core hypothesis underlying these algorithms is to sample fresh hyperparameters based on the history of the search. This makes them much more data-efficient than grid and random search. At the same time, while successive halving begins with a static population, BO starts with only a few (random) picks, and then samples models and hyperparameters based on the cost at those points. In this way, it avoids the problem of pruning potentially good solutions due to limited compute budgets, and provides excellent performance on the exploration-exploitation dilemma [27].

Under the hood, in addition to the cost function as described earlier, BO techniques also rely on a surrogate model and an acquisition function. The surrogate model is used to model the cost (often predictive accuracy, but can also be downstream cost or any other value of interest) as a function of hyperparameter values (i.e. using the samples gathered so far). Models, such as Gaussian Processes (GPs) [28] or Random Forests [29], which can quantify prediction uncertainty are inherently preferred as surrogate models. This surrogate model is subsequently used to create the acquisition function, which guides the optimizer on where to sample next. We note here that the key difference between BO and active learning is that the former is interested in maximizing the pay-out (e.g. in terms of discovering the best model), while the latter concerns itself with learning an accurate mapping at the lowest sampling cost possible. BO, in this context, is therefore the more efficient choice since

we do not wish to learn the entire mapping from hyperparameters to model performance, we only want to discover the best model. For clarity, the overall workflow for BO is given as follows:

1. Define the surrogate model to map between model hyperparameters and performance metric of interest (models that can quantify both the expected value and uncertainty about the estimate such as GPs are popular choices);
2. Define the surrogate model's prior distribution (the prior can be domain-informed or it can be uniformly distributed when no additional information is available);
3. Use Bayes' rule to obtain the posterior, given the function evaluations made so far;
4. Define $\alpha(x)$, the acquisition function. Common choices for the acquisition function include upper confidence bound (UCB), expected improvement (EI) and probability of improvement (PI). Random sampling can also be used to provide a benchmark;
5. Use the acquisition function to decide on the next sample point, based on the updated posterior $\big(x_t = \arg\max_x \alpha(x)\big)\big)$;
6. Update the set of observations;
7. Repeat from step 3, until termination conditions are met.

2.3.5 Application to task-informed forecasters

With this background, we can formulate a Bayesian Optimizer which can directly optimize for downstream cost, or jointly optimize cost and accuracy. In this regard, several algorithms have been proposed over the years to extend the classical BO framework to the multi-objective case [30, 31, 32]. In the following sections, we present the results of applying this framework to tasks in two different domains. However, before doing so, we describe the case studies in greater detail next.

3 Case studies

In this section, we describe how the proposed methodology was applied to two case studies. The first of these pertains to optimizing electricity (peak) demand in a building using battery-inverter systems; the second draws from optimizing pricing and supply chain in the retail sector.

3.1 Energy case

Increasing proliferation of distributed generation via solar PV and electrification of demand (heat pumps, electric vehicles) is causing additional stressors for the distribution (and transmission) grid of electricity. This is further exacerbated by the rapid uptake of renewable energy sources (wind and solar), which is leading to a power system that is increasingly dependent on weather conditions. As a result, consumers and prosumers alike are being incentivized by regulatory and market forces to reduce their electricity demand and shift the remaining demand to points in time when sufficient generation capacity can be guaranteed. A practical manifestation of this is in the form of peak shaving or reduction, which can allow especially large consumers of electricity to reduce their power related costs (e.g. in the form of network utilization fees). In this case study, we explore this idea to see how forecast accuracy affects overall utility for the end consumer or grid, and show how BO can be utilized to discover forecast models that perform much better out-of-the-box (i.e. without recourse to probabilistic forecasting or human judgment).

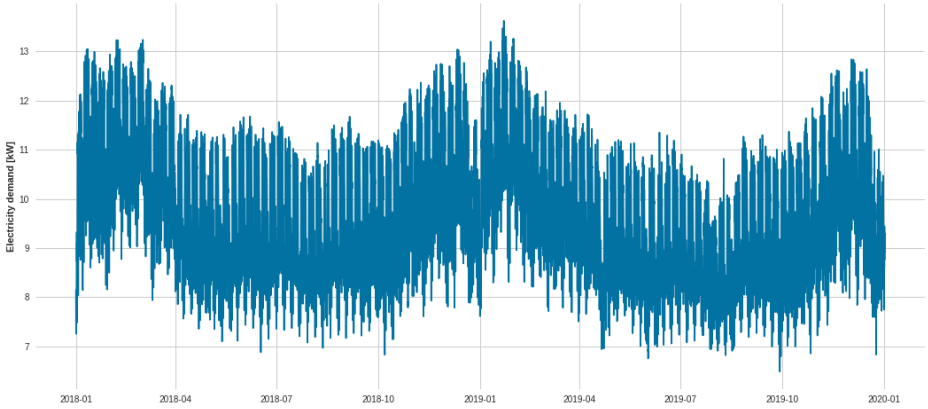

Figure 1: The energy demand time series plotted as a function of time

3.1.1 Data

We begin by assuming that the building follows the Belgian electricity demand time series, as shown in Fig. 1, while the dataframe is shown in Fig. 2 for clarity. Adopting kW as units, we use the observed hourly demand data for 2018 and 2019 for our analysis: the first of these two years is assumed to be available for training the initial models, while the second year is used solely to evaluate the Predict+Optimize workflow on a rolling basis. In this section, we begin by providing a brief snapshot of the most important factors affecting electricity demand. Next, we show how these findings can be leveraged to build forecast models. Finally, we discuss how these forecasts can be used to reduce the electricity peaks by utilizing storage in the form of electrical batteries.

3.1.2 Exploratory data analysis

The time series in question exhibits strong daily, weekly and yearly seasonality, which means that persistence models should provide reasonable baselines. More specifically, the electricity demand is typically higher during the day, following a bimodal distribution, and is particularly low during weekends, holidays and nights. In addition to the calendar features, the electricity demand is also a strong function of the ambient conditions, most notable of which is the outdoor temperature. In a heating-dominated country such as Belgium, the difference temperature variations make to electricity demand can be substantial. These effects are visualized in Fig. 3.

3.1.3 Forecasting

Over the course of the study, we developed several models to forecast the electricity demand time series in question. These include:

1. Persistence models, including both daily and weekly persistence models;
2. Simulated models, including several types of noise (static bias, Gaussian, signal-dependent Gaussian, horizon-dependent Gaussian) injected into the actual time series - these will not be

time	y	Temperature	Day_of_week	Hour_of_day	Holidays
2018-01-01 00:00:00	8.143500	3.50	0	0	1
2018-01-01 01:00:00	7.789000	3.74	0	1	1
2018-01-01 02:00:00	7.470750	2.57	0	2	1
2018-01-01 03:00:00	7.354500	1.96	0	3	1
2018-01-01 04:00:00	7.374250	1.44	0	4	1

Figure 2: A snapshot of the data frame used for the energy case study

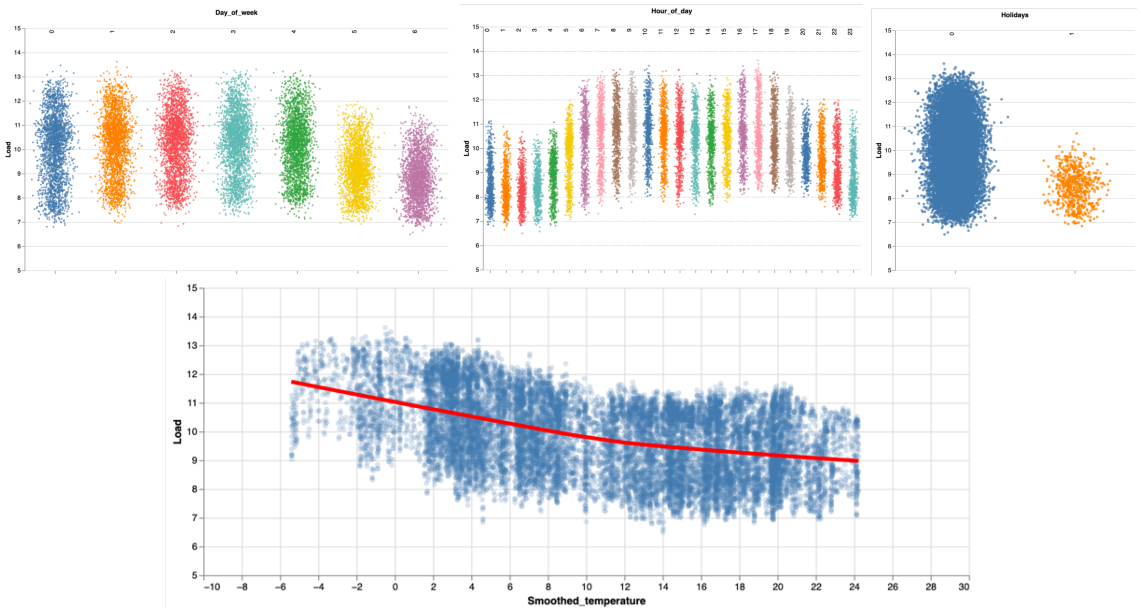

Figure 3: Some factors influencing electricity demand; (top-left): Day of week, (top-right): Hour of day, (bottom-left): Holidays, (bottom-left): Temperature

discussed further in the report;

3. Theta model with weekly seasonality;
4. Linear models (ARX models) incorporating different amounts of historical lags as well as with and without additional covariates (calendar based features including holidays, hour of day, day of week etc., and ambient conditions such as temperature);
5. Elastic Net models which built on the linear models described in the previous point by incorporating both L1 and L2 regularization - note that both ridge and lasso regression are special cases of this model;
6. Tree-based models (LightGBM) incorporating different amounts of historical lags as well as with and without additional covariates (calendar based features including holidays, hour of day, day of week etc., and ambient conditions such as temperature)

All forecast models predicted electricity demand for the next 72 hours (3 days), which was used by the downstream decision-making tool to decide when to charge or discharge the battery in a way that minimized the peak demand violations.

3.1.4 Decision-making

For decision-making to reduce the peak violations, several further choices were necessary. The decision variable is meant to charge or discharge a battery such that it (1) minimizes electricity peak demand, (2) energy usage, and (3) battery-related cycling losses. The decision-making time horizon had a similar horizon of 72 hours. However, employing a receding horizon controller, only the first 24 control actions were actually acted upon. Subsequently, updated forecasts for the next 72 hours were used to update the decision variables. Intra-day recourse on control variables was assumed to be not possible. Where the forecast is inaccurate, the decision will likely be sub-optimal as well. However, in this case, the cost of the forecasts is very obviously asymmetric. A biased over-predictor will naturally lead to fewer grid violations since it will force the battery to be discharged during those times. However, this over-prediction bias, if applied consistently, will not let the controller charge the battery at other times, which will reduce its ability to discharge during subsequent demand peaks. So, while the forecaster can be biased, it still needs to get the timing for this bias right.

In subsequent sections, we focus on these grid violations (defined as the amount the demand exceeds the threshold by) as a proxy for the downstream cost. The decision-making was constrained by pre-defined battery-inverter specifications, i.e. how much and how quickly the battery could be charged or discharged. In the base case, these values were set to roughly half the peak load of approx. 13 kW (4kW/8kWh system). Likewise, the grid constraint was set to 11 kW initially, which proved to be a challenging but managable target, given both the load and the system specifications. Both the battery-inverter specifications and the maximum allowed electricity offtake value were subsequently subjected to a sensitivity analysis to determine how these were affected by the forecast errors.

3.2 Retail case

For many retail platforms (both online and offline), prediction of future sales and optimisation of both inventory stock and discounts can be a great source of cutting costs, and increasing revenues and profits. For instance, retail companies forecast demand of items that can be stocked, and use these forecasts to decide which products to stock and how much, for a given period of time. This

date	item_id	sell_price	wday	snap_CA	snap_TX	snap_WI	sales	special_events	sin_wday	cos_wday	weekend
2013-01-01	FOODS_1_004	1.780	4.0	1.0	1.0	0.0	69.0	1.0	-4.338837e-01	-0.900969	False
2013-01-02	FOODS_1_004	1.780	5.0	1.0	0.0	1.0	81.0	0.0	-9.749279e-01	-0.222521	True
2013-01-03	FOODS_1_004	1.780	6.0	1.0	1.0	1.0	110.0	0.0	-7.818315e-01	0.623490	True
2013-01-04	FOODS_1_004	1.780	7.0	1.0	0.0	0.0	84.0	0.0	-2.449294e-16	1.000000	False
2013-01-05	FOODS_1_004	1.780	1.0	1.0	1.0	1.0	98.0	0.0	7.818315e-01	0.623490	False
...	...	...	...	...	...	...	...	...	...	...	...
2013-12-28	HOUSEHOLD_1_521	0.961	1.0	0.0	0.0	0.0	40.0	0.0	7.818315e-01	0.623490	False
2013-12-29	HOUSEHOLD_1_521	0.961	2.0	0.0	0.0	0.0	48.0	0.0	9.749279e-01	-0.222521	False
2013-12-30	HOUSEHOLD_1_521	0.961	3.0	0.0	0.0	0.0	39.0	0.0	4.338837e-01	-0.900969	False

Figure 4: A snapshot of the data frame used for the retail case study

period can vary with the time of the year, but the stocking decision is typically made days, weeks, or even months in advance (depending on the industry). Once stocked, these retailers can also automate pricing of these items in order to maximise the final revenue and minimize loses from unsold stock. However, it is possible that in addition to increasing sales, a lower price might also cause a decrease in revenues. This can be the case where greater discounts lead to the inventory getting sold out, before reaching the point of profitability. Too high prices, on the other hand, lead to decreased demand and some stock remains unsold at its expiration date (assuming perishable commodities).

Consequently, in this case, we will investigate if incorporating profit increase in the model selection process can reduce this loss in potential revenue. It should be emphasised that conversion rates and elasticity of demand and price can only be derived indirectly in real world settings, i.e. through A/B tests or focus groups etc. To avoid this limitation and to simulate potential result of operating sales with discounts, we made several assumptions (see below).

3.2.1 Data

The data used in this case is from the M5 competition, which was held in 2020. It is available on Kaggle[1]. We did some transformations in order to shape the data in a similar manner to what is used for the energy case study. Furthermore, to reduce sampling bias and intermittent demand, we aggregated sales in different physical stores into a single time series. The final shape of the dataframe used for prediction is shown in Fig. 4.

The set consists of 100 different items that could be predicted and optimised together. We used data for the year 2013. To simplify matters but without loss of generality, we present results for one product in this report.

3.2.2 Forecasting future sales

The same forecast models as those discussed in the energy section are utilizer for this case as well, including: persistence baselines, Theta model, OLS linear regression, Elastic Net, and LightGBM. The models utilize historic data (i.e. lags), as well as calendar features as covariates to produce forecasts. To differentiate from the energy case, the forecast quality is measured using root mean

[1] https://www.kaggle.com/competitions/m5-forecasting-accuracy/overview

squared error (RMSE). As before, the prediction is set to 3 days days ahead. Unlike in the energy case, the data is sampled daily instead of hourly.

3.2.3 Decision-making

In this case, the control action consists of two steps: decision on setting the (1) discount for, and (2) stock of an item, according to the forecast sales.

Optimization of discounts

After creating several forecasts, we run an optimisation algorithm to maximise potential profit. The algorithm is looking for the best discount for each item, following a set of rules and assumptions:

1. We make daily predictions for each item and use this to calculate total revenue for the day (price/item * prediction for number of items sold);
2. Profit is calculated as the money we earn minus the cost of the item for us. Cost of each item is set to 20% of its base (undiscounted) price;
3. We implement a discount, in the range of 0 and 60%, for each individual item to increase sales;
4. Effect of a discount is pre-set: we get 3% increase in sales for each 1% discount;
5. The stock inventory sets a limit on the maximum amount of items that can be sold during one day.

This workflow is executed for the forecasts from all models to get recommended actions.

Optimization of stocking Inventory

The demand prediction (incorporating discounts), N, defines the stock inventory (i.e. we set the stock inventory to N). Once this has been implemented:

1. For the case where the demand is higher than the prediction, the sales will be limited to N;
2. When demand is lower than the prediction, the unsold inventory is assumed to go bad after a predefined time period, and is not available for sale anymore. As a result, the cost (20%) will be deducted from the final profit;

Here as well, we will repeat the procedure for each predictive model.

3.2.4 Deriving the final profit increase

After applying discounts and stocking items in the inventory we can simulate how the actual demand would have been affected by the operations and how the different models would have contributed to the profit increase. In order to get the final profit increase, we take the factual sales for the same period, implement our discounts, according to the optimal decision for each model. We also adjust sales, according to the same pre-set rule: 3% increase in sales for each 1% discount. Now, knowing factual sales and how they would look like with the discount, we can calculate the final profit for each model. The stocking inventory from the previous step comes into play here, because we can only sell as much as we have in stock, and the cost of the items that haven't been sold will be deducted from the final profit. In subsequent sections, this potential for profit increase by setting discounts and inventory will refer to the downstream utility of the model.

4 Implementation details

To implement the end-to-end BO framework, we experimented with several different Python implementations for the three core tasks: forecasting, decision-making and BO itself. These are described in this section:

1. **Forecasting.** We developed the forecasters using several different packages, before settling on the Darts package from Unit8 [33]. This library acts as a wrapper package with the ability to call standard function approximation techniques implemented in other libraries such as the Scikit-Learn models [34] and LightGBM [35]. The use of Darts unifies primitives such as $model.fit$ and $model.predict$ routines, besides providing the framework to compare models developed using different libraries. As such, it helped us test the generalization capabilities of our code, and its robustness to the use of different libraries or software packages when making forecasts.

2. **Downstream decision-making.** To implement the downstream decision-making, we also experimented with several libraries. Since the energy case could be posed as a linear program, we used the Python library PuLP [36] using CBC [37] as the solver. For the retail case, we used both the open-source derivative-free library Nevergrad [38] and Optuna [39]. Both of these libraries provide several routines for black-box optimization, which we selected based on their downstream performance.

3. **Bayesian Optimization.** To implement the hyperparameter search, we utilized Optuna. This library provides several implementations for BO as well as random sampling and evolutionary algorithms (e.g. CMA-ES [40]). More concretely, we used the Tree Parzen Estimator (TPE) sampler [41] to select the next samples during the hyperparameter search. This sampler can be used for multi-objective optimization, and has time complexity of $\mathcal{O}(dn \log n)$, where d is the dimension of the search space, and n is the number of finished trials. This time complexity for BO is as opposed to $\mathcal{O}(n^3)$ for CMA-ES, which is much more intensive in the amount of trials.

5 Results

In this section, we present the most important results from applying the proposed methodology to the two use cases.

5.1 Energy case

For the energy case, we begin by describing the results of using the proposed BO framework to select models that optimize for accuracy, utility or both. We do so by initially focusing on hyperparameter tuning, before looking into the entire end-to-end CASH problem formulation, both for the single and multi-objective cases. This already provides us with a view of how different factors influence the search. Subsequently, we consider different variations to important factors used in the base case to investigate their impact on downstream results. These include the choice of sampler function and the sensitivity to problem formulation.

5.1.1 BO for hyperparameter tuning

In the first experiment, we limit our attention to only a single model, i.e. we treat only the hyperparameter optimization aspect of the CASH problem. To do so, we choose the LightGBM model

Hyperparameter	Value range
Input lags	(1, 672)
Maximum depth	(1, 12)
Number of estimators	(100, 10000)
Learning rate	(0.01, 0.3)
Number of leaves	(20, 3000)
Strength of L1 regularization	(0, 100)
Strength of L2 regularization	(0, 100)

Table 1: Choice of hyperparameters for LightGBM models

because of its documented excellent performance in tabular tasks [42, 43]. We optimize several hyperparameters for the LightGBM model, including the maximum depth, number of estimators and leaves per tree, the learning rate and the strength of regularization, as mentioned in Table 1. These factors influence both the complexity of the function the model can learn, and how quickly it can learn it. At the same time, the time complexity of TPE grows with the number of hyperparameters so irrelevant hyperparameters should be avoided.

Running the Predict+Optimize loop for the first three months of 2019 (see Fig. 1), we obtain the results shown in Fig. 5. The three panels of Fig. 5 confirm our hypothesis that when the predefined constraints place asymmetric costs on over- and under-prediction, the search for most accurate and useful models diverges in two different directions. Consequently, we note that the most accurate point forecast model leads to much higher downstream costs than the downstream cost-optimized model. This latter model, on the other hand, traverses the hyperparameter space in a very different direction, leading to a model that has much better downstream performance than the most accurate model, at the cost of lower accuracy. The right panel shows the Pareto front which can be obtained using multi-objective BO that enables us to obtain several models which jointly optimize both accuracy and downstream costs. The shading of the markers on the graph highlights three additional aspects, which we discuss next:

1. The Pareto front is marked in red on all three panels - for the single objective cases, this is a single forecaster as expected, while for the multi-objective case it includes several forecasters, each of which is non-dominated. As discussed earlier, the most accurate model is not necessarily the most useful.

2. The figure shows three baselines (marked as dark, filled circles), including the naive daily and weekly persistence models, as well as the perfect model case (with no forecast error). It is obvious that the models under consideration learn from data to outperform the persistence models in terms of both accuracy and utility. What is more remarkable is that, despite still being rather inaccurate, the lowest cost model has actually converged to a utility that is only marginally worse than if we were to utilize the observation data (instead of the forecasts). Expending a lot of computational resources or algorithmic effort to discover a more accurate model, compared to the one discovered by BO, would therefore not significantly further improve performance.

3. The markers in the scatter plots are shaded according to their order of sampling (samples drawn later in the optimization process are shaded in a darker color). This shows that, as anticipated, the BO nudges the draw of hyperparameters in a way that quickly converges towards a region which optimizes performance according to the defined criterion.

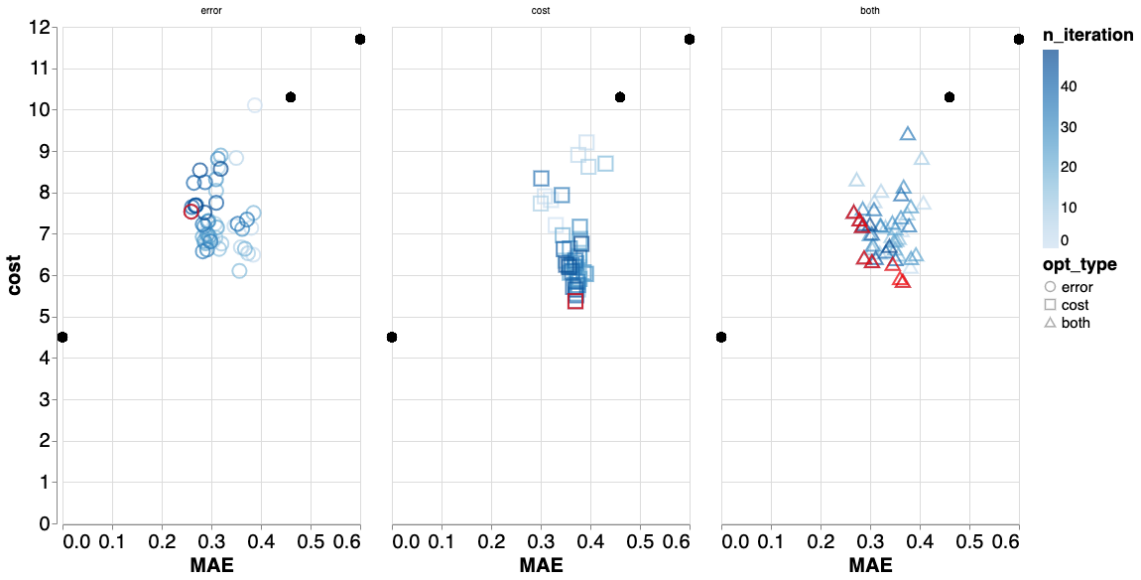

Figure 5: BO with TPE sampler for (left) accuracy, (middle) cost, (right) both accuracy and cost; the 'best' model or models (Pareto front) are marked in red; the baseline models (daily and weekly persistence forecasts, ideal forecast) are marked as solid black circles

5.1.2 Interpreting BO feature importances

A useful feature of BO, as discussed in an earlier section, is that the surrogate models also provide some insights into the search process, i.e. which hyperparameters are driving improvements in the model performance. This applies equally to both models optimized for accuracy and those optimized for cost (or any combination of the two). These results are shown in Fig. 6, which shows that the most important hyperparameters differ substantially for different objective functions. For the most accurate model, L1 regularization strength and learning rate are the most important. For the most useful model, on the other hand, the lags and maximum depth of the model are most important. We found the dual-objective trained models to instead exhibit more diffuse hyperparameter importances. Notably, for the utility dimension, the most important hyperparameters include the historical lags, the L1 regularization strength, and the maximum depth of the model. Note that these do not represent which features are most important in making the predictions, it highlights instead which hyperparameters are the most important in nudging the model in one direction or a different one.

These results show clearly that BO learns to focus on different aspects of the hyperparameter space to find the models that optimize the stated objective function.

5.1.3 Benchmarking BO against random search

In contrast to Fig. 5, Fig. 7 shows results of applying random sampling to the same problem setup. Unlike for BO, there is no obvious sampling bias here, which, as expected, reflects the random nature of the draws. As such, even though both algorithms can (and should) theoretically converge to comparable solutions, the BO algorithm does so at a fraction of the computational cost of the random sampler. The discussion section provides more details on this aspect.

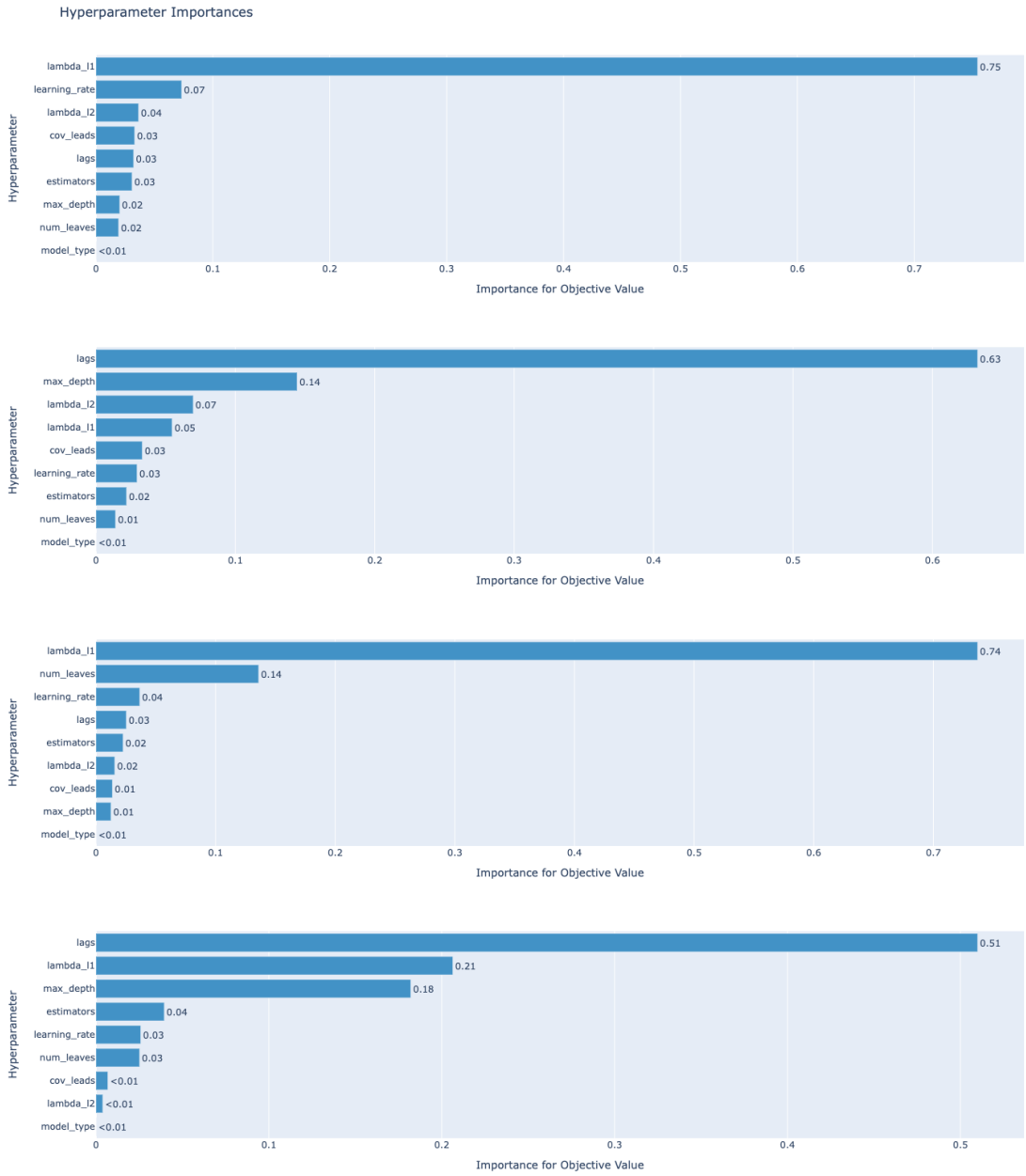

Figure 6: Hyperparameter importances obtained through the TPE sampler with BO for different model configurations and objectives; (top to bottom): hyperparameter importances for the model trained on accuracy objective; hyperparameter importances for the model trained on downstream cost objective; hyperparameter importances for the model trained on both objectives, visualized for accuracy; hyperparameter importances for the model trained on both objectives, visualized for utility

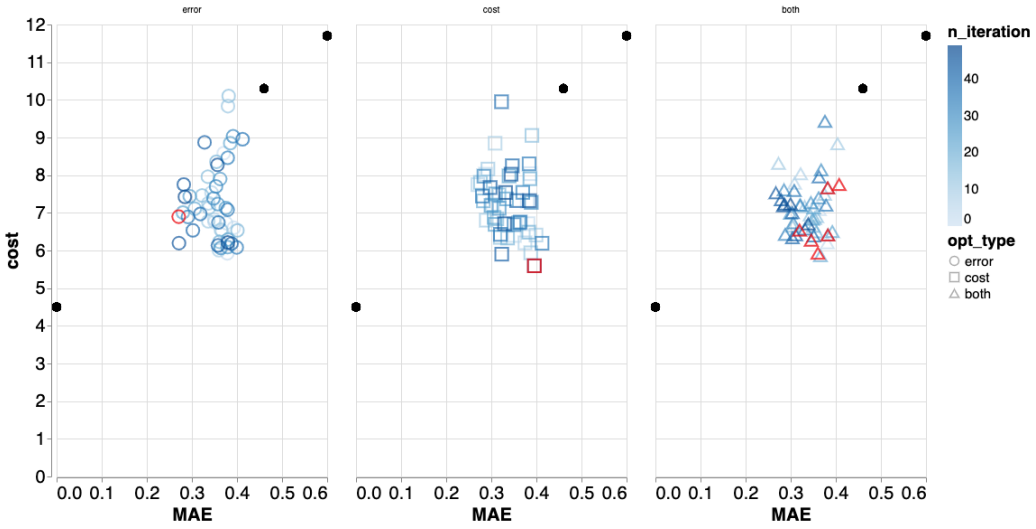

Figure 7: Results of applying random sampling for (left) accuracy, (middle) cost, (right) both accuracy and cost

5.1.4 BO for extended CASH

In the next experiment, we expanded our search space to also include the model type. This meant searching over four different model types, each with its own hyperparameters:

1. LightGBM models retained the same search space as from the previous experiments;
2. OLS linear regression models were introduced which used only two hyperparameters: historical lags and future values of covariate forecasts;
3. Elastic Net models were tested with the same historic and future value lags as LightGBM and linear regression; additionally the L1 and L2 regularization strength were also used as hyperparameters;
4. Theta model with hyperparameters including the seasonality period (daily or weekly) and the theta parameter.

Running the entire CASH workflow on just optimizing for accuracy provides an insight into the behaviour different models exhibit. This is explored in Fig. 8 which shows the overall cost-accuracy scatter plot as well as four panels disaggregated by model type. From the figure, it is obvious that the model's downstream performance improves to a certain extent with improving accuracy. However, there appears an inflection around an MAE of 0.4. Improvements to model accuracy beyond this point seem to lead to worsening of downstream performance. It is also evident from the disaggregated panels that the inflection is due to the behaviour of different model types (i.e. it marks the transition between Elastic Net models, and the linear and LightGBM models). Running the proposed dual-objective BO can address this challenge by also forcing LightGBM models to converge closer to lower costs.

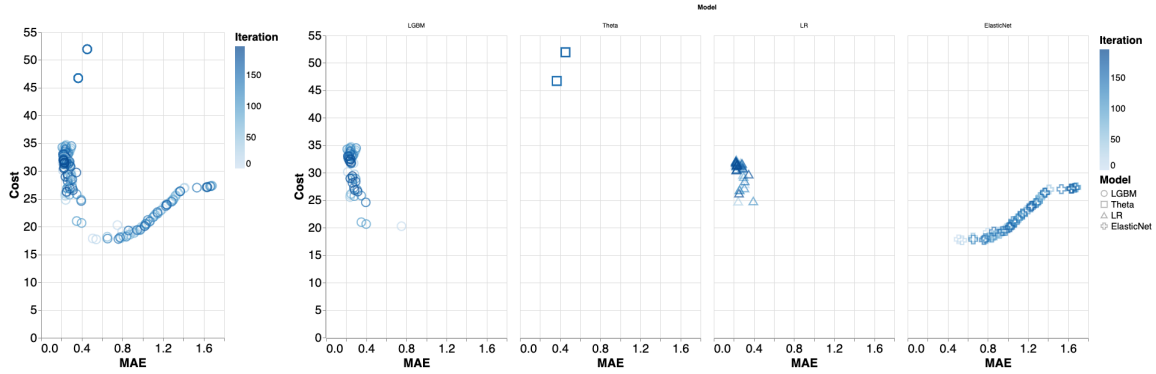

Figure 8: The cost-accuracy trade-off for the case where peak prediction is extremely important, both aggregated (left) and disaggregated by model type (right panels)

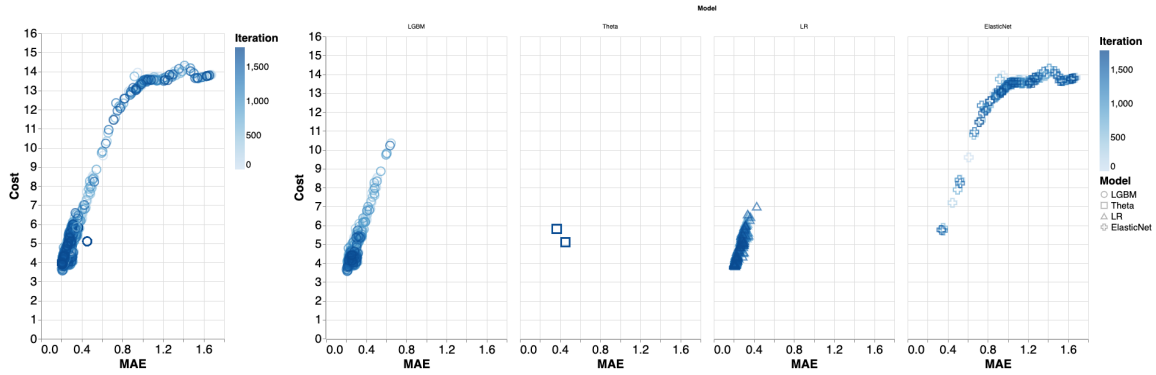

Figure 9: The cost-accuracy trade-off for the case where peak prediction is balanced by downstream storage concerns, both aggregated (left) and disaggregated by model type (right panels)

5.1.5 Sensitivity to grid and storage constraints

In the case we have examined so far, predicting the peaks correctly is absolutely vital. If we relax the peak reduction constraints to the extent that other decision-making concerns such as battery costs also play a role in the downstream decision making process, we see a different dynamic emerging. This is shown in Fig. 9, where the cost and accuracy are very well correlated for both the linear and LightGBM models. For the Elastic Net models likewise, we see that the model eventually improves simultaneously on both accuracy and cost despite struggling with selecting the appropriate hyperparameters.

In addition to modifying the constraints, we also changed the specifications of the electrical battery (i.e. storage) by a factor of 2 in both directions. Common wisdom, typically in the energy sector, goes that if we had unlimited flexibility via storage, forecast accuracy would be irrelevant. In this case, we did not observe this phenomenon. Instead, a largely similar effect as seen in Figs. 8 and 9 played out.

Hyperparameter	Value range
Input lags	(1, 28)
Maximum depth	(1, 12)
Number of estimators	(100, 10000)
Learning rate	(0.01, 0.3)
Number of leaves	(20, 3000)
Strength of L1 regularization	(0, 100)
Strength of L2 regularization	(0, 100)

Table 2: Choice of hyperparameters for LightGBM models, Retail Case

5.2 Retail case

This case explores the potential for an actual profit increase due to the different forecast models. This increase will then be compared against the profit that could actually be realized, i.e. the one obtained without any manipulations. Here, as before, we begin by using the proposed BO framework for both single and multi-objective optimization, with single objectives being either RMSE or profit increase.

5.2.1 BO for hyperparameter tuning

As before, for the first experiment, we used only a LightGBM model, and tested its performance along several hyperparameters such as number of historical lags, number of estimators and leaves, learning rate, max depth, and strength of L1 and L2 regularization, as mentioned in Table 2.

After running the BO for all three objectives (accuracy, profit, accuracy+profit), we obtain the results shown in Fig. 10. To assess the performance of this model against bounds provided by benchmarks, we are using two persistence models (naive daily and naive weekly), as well as factual data. The results show that the most accurate models from RMSE perspective, don't necessary guarantee the very best results for profit increase. Moving to an optimizer, based on increase in the downstream value, we get an improvement in the both dimensions (RMSE and Profit increase). From the BO path in Fig. 10, we can see how the RMSE-driven optimizer (the left panel) converges to the region with the lowest error, taking a route that doesn't lead to higher downstream value though. At the same time, the downstream-value driven algorithm (the middle panel) pushing the results towards higher profit increase. Because profit increase in the case is directly connected with the quality of a forecast, optimisation for a higher profit increase also provides solutions with lower error. The right panel in the graph represents the multi-objective optimization algorithm, and the best solution (marked red) lies on the front that is moving towards top left corner. This solution is similar to the downstream-value one.

5.2.2 Interpreting BO feature importance

In Fig. 11, as for the energy case, we can see how different features gain (or lose) importance depending on the target value for optimisation. While in all the cases lags are by far the most important hyperparameter, when optimising for RMSE, L1 and L2 regularisation gain more importance. And with profit increase we can see max depth and learning rate gaining more weight when picking the winning model as well. It is also interesting to see how the importance of historical lags is increasing with multi-objective optimization, comparing to the single-objective cases, when different model hyperparameters have more relative weight.

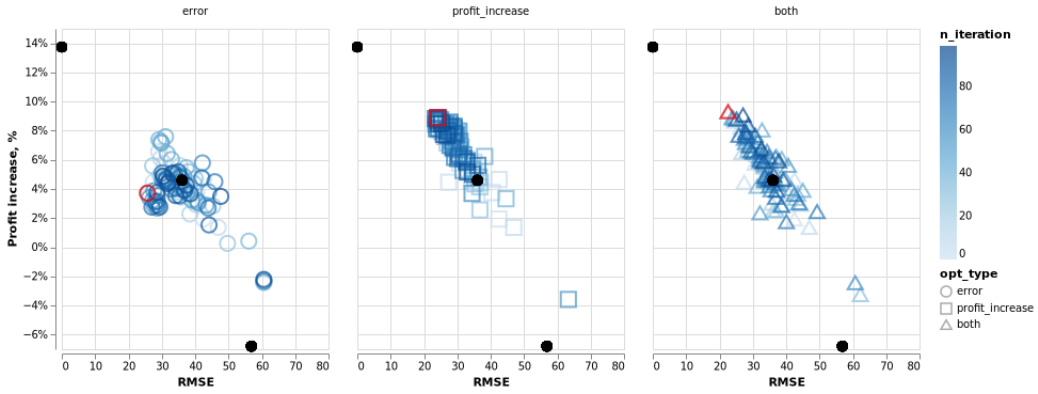

Figure 10: BO with TPE sampler for (left) accuracy, (middle) profit increase, (right) both accuracy and profit increase. Model is limited to LGBM only. The Black dots represent persistence models, from left to right: fact, naive weekly and naive daily.

5.2.3 BO for extended CASH

For this experiment we include more model architectures into the hyperparameter search space:

1. LightLGBM models, similar to the ones used in the previous step;
2. RandomForest models with number of lags, maximum depth and number of estimators as hyperparameters;
3. ElasticNet models with lags, and L1 and L2 regularization strength.

As seen in Fig. 12, by making the hyperparameter space model-agnostic we manage to increase the speed of convergence, as some of the models converged to their best performing value in fewer iterations (see Fig. 13). One final experiment in this context was to run the same Predict+Optimize problem with multivariate TPE algorithm (see Fig. 14). This configuration of the TPE algorithm often outperforms univariate TPE [44], which does not account for interactions between different hyperparameters. In this case, it led to slightly better results compared to regular TPE by exploring more options around optimal values in the same number of iterations.

5.2.4 Benchmarking BO against random search

Finally, in the last experiment we run a similar Predict+Optimize loop, now with Random sampling as the hyperparameter tuning strategy. Fig. 15 showcases that the final results the algorithm converged to are rather similar to the previous runs with Bayesian approach (Fig. 12 and Fig. 14). At the same time, getting there takes more iterations, and that number rapidly increases with increasing size and dimensionality of the hyperparameter space.

6 Discussion and conclusions

The method discussed in this paper, i.e. utilizing Bayesian Optimization to jointly optimize for model accuracy and downstream performance, lies between two opposite ends of the spectrum that have

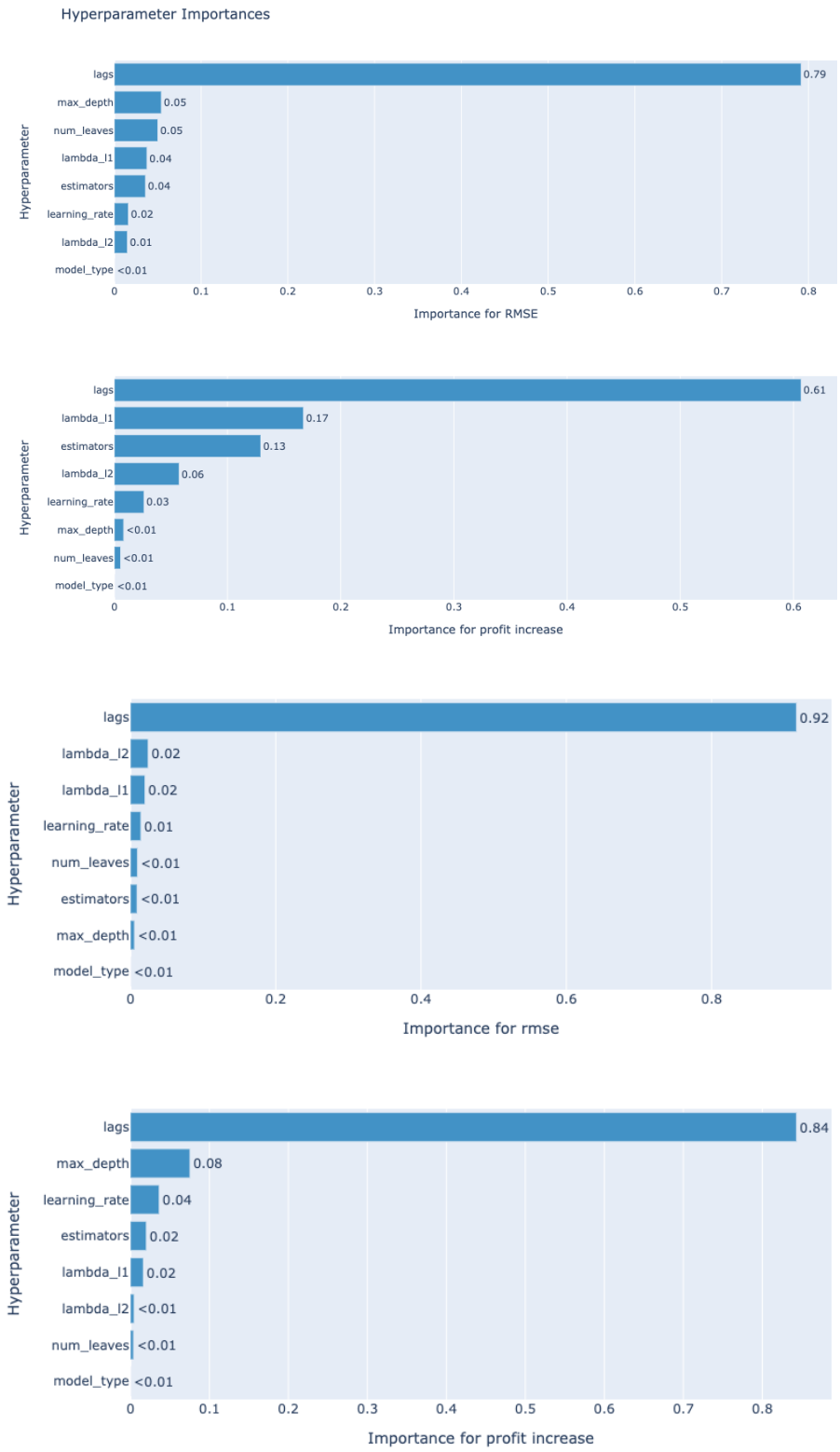

Figure 11: Hyperparameter importances obtained through the TPE sampler with BO for LightGBM and objectives; (top to bottom): hyperparameter importances for the model trained on accuracy objective; hyperparameter importances for the model trained on profit increase objective; hyperparameter importances for the model trained on both objectives, visualized for accuracy; hyperparameter importances for the model trained on both objectives, visualized for utility

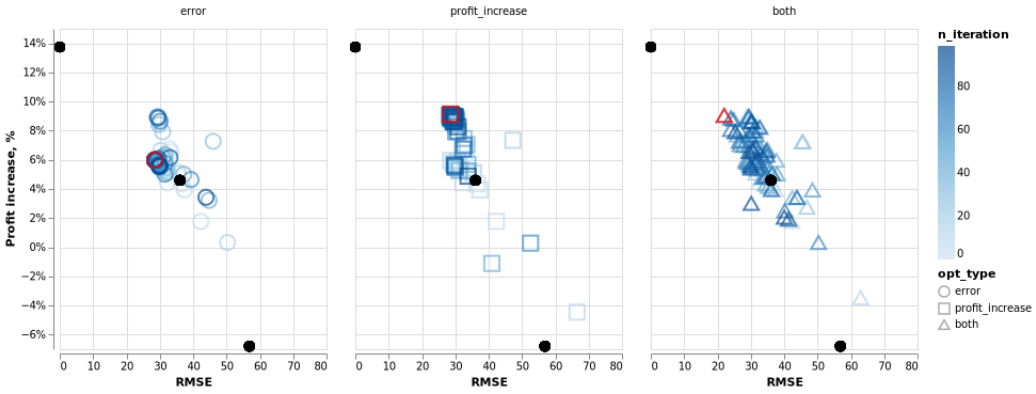

Figure 12: BO with TPE sampler for (left) accuracy, (middle) profit increase, (right) both accuracy and profit increase. The Black dots represent persistence models, from left to right: fact, naive weekly and naive daily.

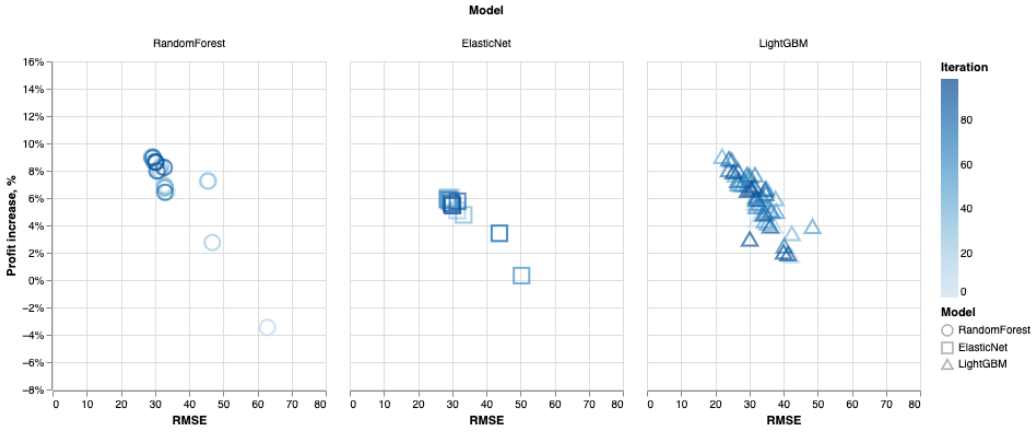

Figure 13: BO with TPE sampler for both accuracy and profit increase split by models architecture

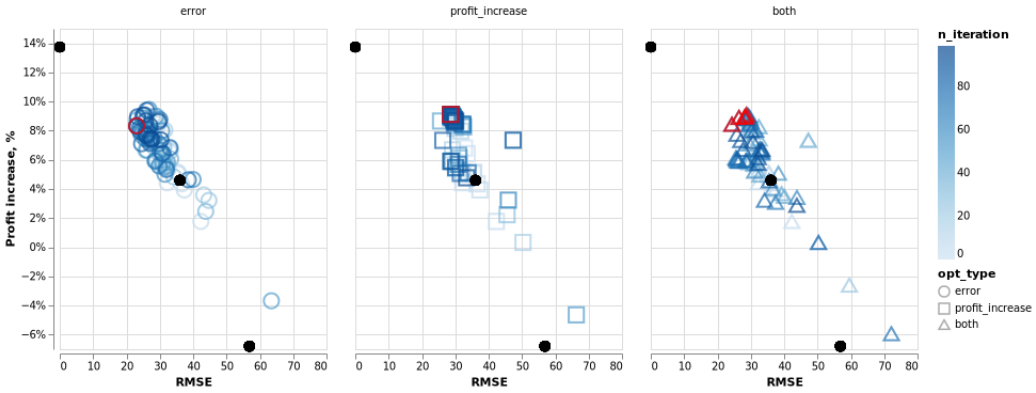

Figure 14: BO with multivariate TPE sampler for (left) accuracy, (middle) profit increase, (right) both accuracy and profit increase. The Black dots represent persistence models, from left to right: fact, naive weekly and naive daily.

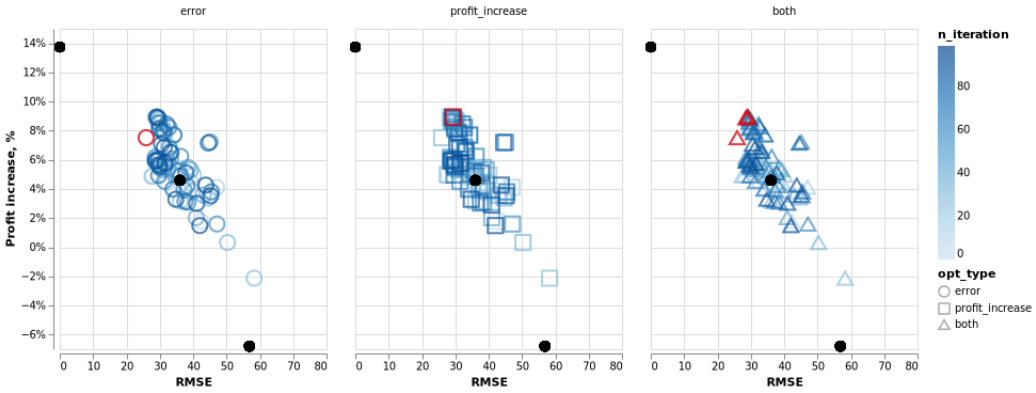

Figure 15: BO with Random sampler for (left) accuracy, (middle) profit increase, (right) both accuracy and profit increase. The Black dots represent persistence models, from left to right: fact, naive weekly and naive daily.

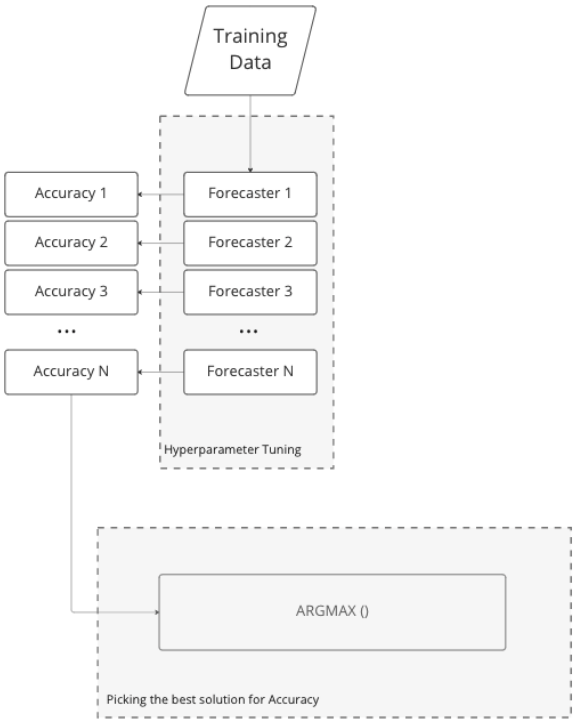

Figure 16: The classical machine learning model creation loop

been explored in literature. On the one extreme lies the classical machine learning paradigm which focuses on accuracy-related metrics, and can be tweaked using human expertise for risk-constrained decision-making. On the other extreme lie recent methods which propose to create models that directly integrate downstream costs into their training process. The former requires considerable human domain knowledge and discipline, both of which are seldom, if ever, guaranteed in practical settings. The latter is computationally intensive to the point of being intractable for all but the simplest of downstream tasks. Figs. 16 and 17 show these concepts respectively.

The proposed method (visualized in Fig. 18 in a simplified form) leverages the insight that all machine learning models, including those that are trained directly using downstream costs, require considerable hyperparamter tuning. This is, in itself, a computationally expensive task. By integrating the task information at this level, we obtain models that are not just accurate and useful, but also remain computationally tractable. We have shown this to be true through several experiments in different domains, including energy and retail. More specifically, our results demonstrate that this method has sufficient discriminative power to choose the most accurate model when accuracy is well correlated with downstream utility. However, in cases where this does not hold, the proposed method quickly discovers alternative forecasters that can prioritize downstream task utility or return the Pareto front that optimizes for both simultaneously.

Besides being computationally tractable and advantageous in terms of utility, the method is completely model-agnostic. We have demonstrated this by applying it to the full CASH (Combined Algorithm Selection and Hyperparameter Optimization) problem formulation. Consequently, it can

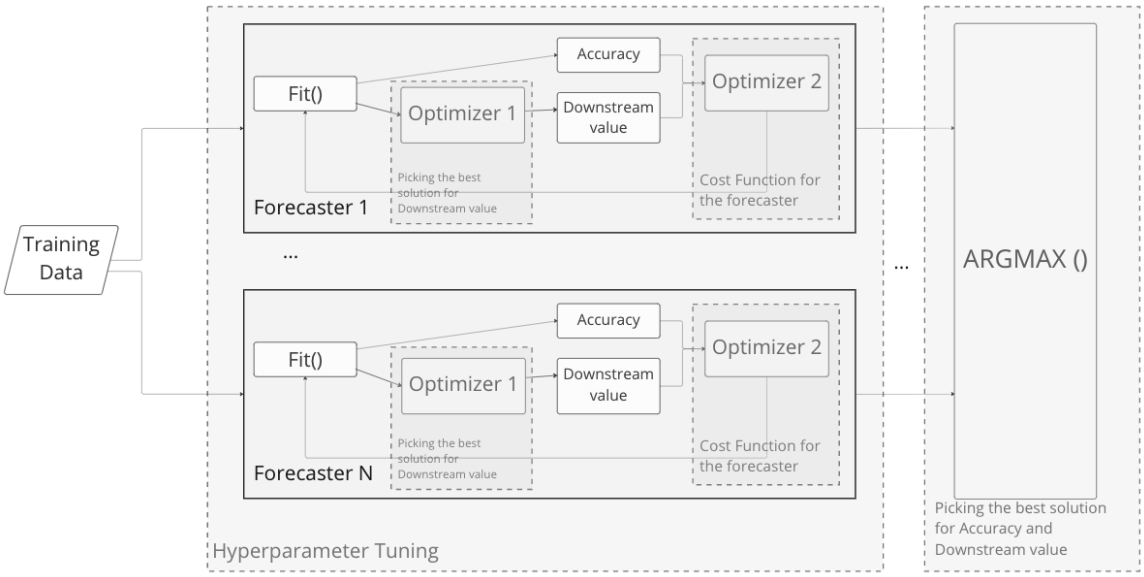

Figure 17: Running Predict+Optimize at every iteration to train a machine learning model

be used with any function approximation technique ranging from very simple statistical methods to state of the art boosting techniques (or even deep neural networks). With prior techniques that integrate downstream costs directly into the training step, this is naturally not possible (since every technique has its own specific training routine). At the same time, the Bayesian Optimization workflow also results in a hyperparameters importance matrix that can be used to quickly understand whether models built for accuracy and utility prioritize the same or different architectures and hyperparameters.

Finally, we note that TPE sampling significantly outperforms naive counterparts such as random or grid search, both in terms of the quality of solutions it discovers and the pace at which it discovers them. This is shown in Fig. 19, which shows the profit increase discovered by the best model as a function of the iteration number. To ensure statistical significance, we ran the same simulation with ten random seeds to initialize both TPE and random sampling, which forms the basis for the confidence intervals in the plot. The results clearly indicate BO outperforms random sampling significantly over time. Further gains are possible using very recent innovations, such as multivariate TPE sampling, but these remain small for now. Future algorithmic innovations will undoubtedly open the door to increasing incorporation of downstream tasks in creation of forecasters.

In terms of dissemination, preliminary work carried out over the course of 2021 and 2022 was presented at several prestigious venues, including at the ISF 2022, as well as in meetings at KU Leuven and National University of Singapore, besides others. As potential next steps, we intend to (1) submit our findings to the IJF, and (2) present our key findings at other venues of interest including topical conferences in data science and operations research etc.

Acknowledgments

Hussain Kazmi and Maria Paskevich gratefully acknowledge the IIF-SAS Annual award.

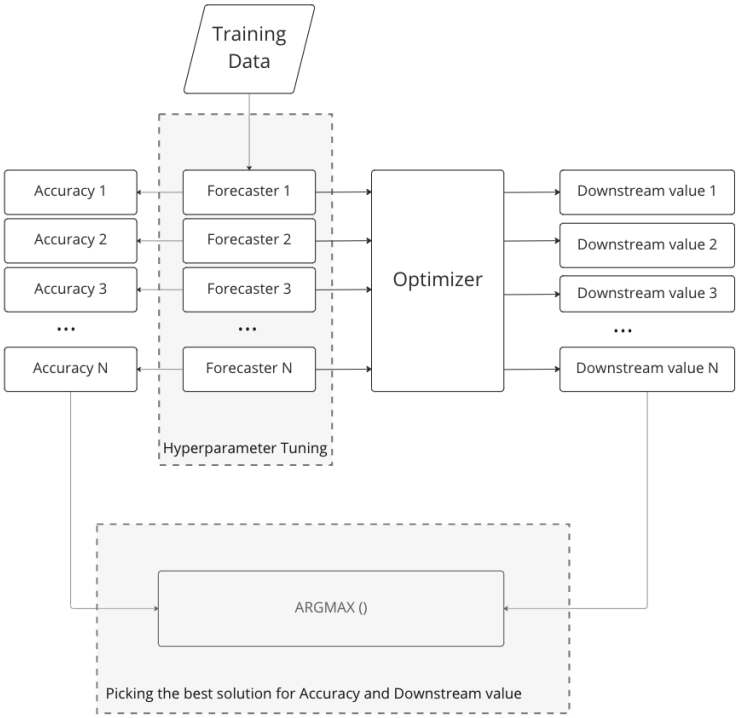

Figure 18: The proposed BO framework for forecast models that co-optimizes accuracy and downstream cost function in a model-agnostic and computationally tractable manner

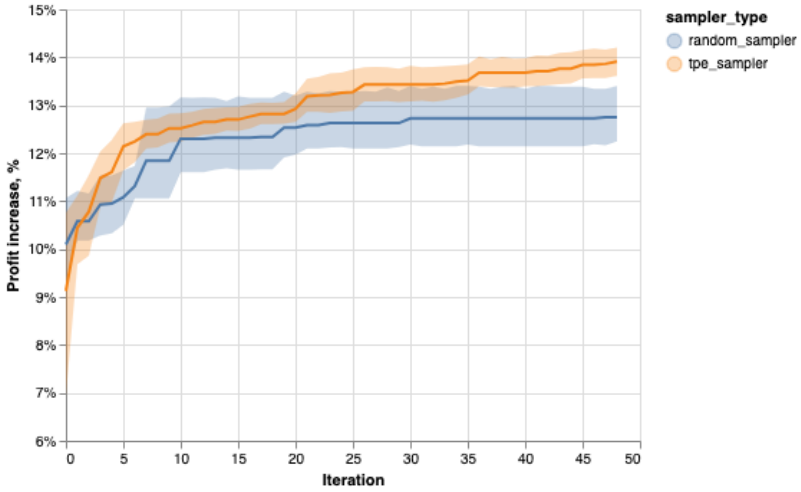

Figure 19: Performance of TPE and random sampling on the retail task, averaged over 10 runs

References

[1] Glenn W Brier et al. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[2] Hansika Hewamalage, Klaus Ackermann, and Christoph Bergmeir. Forecast evaluation for data scientists: Common pitfalls and best practices. *arXiv preprint arXiv:2203.10716*, 2022.

[3] Shari De Baets and Nigel Harvey. Using judgment to select and adjust forecasts from statistical models. *European Journal of Operational Research*, 284(3):882–895, 2020.

[4] Robert Fildes and Paul Goodwin. Against your better judgment? how organizations can improve their use of management judgment in forecasting. *Interfaces*, 37(6):570–576, 2007.

[5] Tilmann Gneiting and Matthias Katzfuss. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 1:125–151, 2014.

[6] Christoph Bergmeir, Frits de Nijs, Abishek Sriramulu, Mahdi Abolghasemi, Richard Bean, John Betts, Quang Bui, Nam Trong Dinh, Nils Einecke, Rasul Esmaeilbeigi, et al. Comparison and evaluation of methods for a predict+ optimize problem in renewable energy. *arXiv preprint arXiv:2212.10723*, 2022.

[7] Mahdi Abolghasemi and Richard Bean. How to predict and optimise with asymmetric error metrics. *arXiv preprint arXiv:2211.13586*, 2022.

[8] Akylas Stratigakos, Simon Camal, Andrea Michiorri, and Georges Kariniotakis. Prescriptive trees for integrated forecasting and optimization applied in trading of renewable energy. *IEEE Transactions on Power Systems*, 37(6):4696–4708, 2022.

[9] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[10] Pablo Montero-Manso, George Athanasopoulos, Rob J Hyndman, and Thiyanga S Talagala. Fforma: Feature-based forecast model averaging. *International Journal of Forecasting*, 36(1):86–92, 2020.

[11] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Bernie Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, et al. Neural forecasting: Introduction and literature overview. *arXiv preprint arXiv:2004.10240*, 2020.

[12] Souhaib Ben Taieb and Rob Hyndman. Recursive and direct multi-step forecasting: the best of both worlds. Technical report, Monash University, Department of Econometrics and Business Statistics, 2012.

[13] Xin Guo, Bas van Stein, and Thomas Bäck. A new approach towards the combined algorithm selection and hyper-parameter optimization problem. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2042–2049. IEEE, 2019.

[14] Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-WEKA: Automatic model selection and hyperparameter optimization in WEKA. In *Automated machine learning*, pages 81–95. Springer, Cham, 2019.

[15] Tianyu Mu, Hongzhi Wang, Chunnan Wang, Zheng Liang, and Xinyue Shao. Auto-CASH: A meta-learning embedding approach for autonomous classification algorithm selection. *Information Sciences*, 591:344–364, 2022.

[16] PM Lerman. Fitting segmented regression models by grid search. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 29(1):77–84, 1980.

[17] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[18] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[19] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[20] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial intelligence and statistics*, pages 240–248. PMLR, 2016.

[21] Manoj Kumar, George E Dahl, Vijay Vasudevan, and Mohammad Norouzi. Parallel architecture and hyperparameter search via successive halving and classification. *arXiv preprint arXiv:1805.10255*, 2018.

[22] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, Nando De Freitas, et al. Bayesian optimization in high dimensions via random embeddings. In *IJCAI*, volume 13, pages 1778–1784, 2013.

[23] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial intelligence and statistics*, pages 528–536. PMLR, 2017.

[24] Aaron Klein, Stefan Falkner, Numair Mansur, and Frank Hutter. Robo: A flexible and robust bayesian optimization framework in python. In *NIPS 2017 Bayesian optimization workshop*, pages 4–9, 2017.

[25] Artur Souza, Luigi Nardi, Leonardo B Oliveira, Kunle Olukotun, Marius Lindauer, and Frank Hutter. Bayesian optimization with a prior for the optimum. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 265–296. Springer, 2021.

[26] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile bayesian optimization package for hyperparameter optimization. *J. Mach. Learn. Res.*, 23:54–1, 2022.

[27] Mohan Yogeswaran and SG Ponnambalam. Reinforcement learning: exploration–exploitation dilemma in multi-agent foraging task. *Opsearch*, 49:223–236, 2012.

[28] Yee-Fun Lim, Chee Koon Ng, US Vaitesswar, and Kedar Hippalgaonkar. Extrapolative bayesian optimization with gaussian process and neural network ensemble surrogate models. *Advanced Intelligent Systems*, 3(11):2100101, 2021.

[29] Difan Deng and Marius Lindauer. Searching in the forest for local bayesian optimization. In *ECMLPKDD Workshop on Meta-Knowledge Transfer*, pages 38–50. PMLR, 2022.

[30] Nazan Khan, David E Goldberg, and Martin Pelikan. Multi-objective bayesian optimization algorithm. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 684–684, 2002.

[31] Samuel Daulton, Sait Cakmak, Maximilian Balandat, Michael A Osborne, Enlu Zhou, and Eytan Bakshy. Robust multi-objective Bayesian optimization under input noise. In *International Conference on Machine Learning*, pages 4831–4866. PMLR, 2022.

[32] Paulo Paneque Galuzio, Emerson Hochsteiner de Vasconcelos Segundo, Leandro dos Santos Coelho, and Viviana Cocco Mariani. MOBOpt—multi-objective bayesian optimization. *SoftwareX*, 12:100520, 2020.

[33] Julien Herzen, Francesco Lässig, Samuele Giuliano Piazzetta, Thomas Neuer, Léo Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, et al. Darts: User-friendly modern machine learning for time series. *Journal of Machine Learning Research*, 23(124):1–6, 2022.

[34] Oliver Kramer and Oliver Kramer. Scikit-learn. *Machine learning for evolution strategies*, pages 45–53, 2016.

[35] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[36] Stuart Mitchell, Michael OSullivan, and Iain Dunning. PuLP: a linear programming toolkit for python. *The University of Auckland, Auckland, New Zealand*, 65, 2011.

[37] John Forrest and Robin Lougee-Heimer. CBC user guide. In *Emerging theory, methods, and applications*, pages 257–277. INFORMS, 2005.

[38] Pauline Bennet, Carola Doerr, Antoine Moreau, Jeremy Rapin, Fabien Teytaud, and Olivier Teytaud. Nevergrad: black-box optimization platform. *ACM SIGEVOlution*, 14(1):8–15, 2021.

[39] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[40] Ilya Loshchilov and Frank Hutter. CMA-ES for hyperparameter optimization of deep neural networks. *arXiv preprint arXiv:1604.07269*, 2016.

[41] Yoshihiko Ozaki, Yuki Tanigaki, Shuhei Watanabe, and Masaki Onishi. Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In *Proceedings of the 2020 genetic and evolutionary computation conference*, pages 533–541, 2020.

[42] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[43] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data? *arXiv preprint arXiv:2207.08815*, 2022.

[44] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pages 1437–1446. PMLR, 2018.