ART: A machine learning Automated Recommendation Tool for synthetic biology

Tijana Radivojević,[†,‡] Zak Costello,[¶,†,‡] Kenneth Workman,[†,‡,§] and Hector Garcia Martin[*,¶,†,‡,‖]

†*DOE Agile BioFoundry, Emeryville, CA, USA.*

‡*Biological Systems and Engineering Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA.*

¶*Biofuels and Bioproducts Division, DOE Joint BioEnergy Institute, Emeryville, CA, USA.*

§*Department of Bioengineering, University of California, Berkeley, CA, USA*

‖*BCAM, Basque Center for Applied Mathematics, Bilbao, Spain.*

E-mail: hgmartin@lbl.gov

Abstract

Synthetic biology allows us to bioengineer cells to synthesize novel valuable molecules such as renewable biofuels or anticancer drugs. However, traditional synthetic biology approaches involve ad-hoc engineering practices, which lead to long development times. Here, we present the Automated Recommendation Tool (ART), a tool that leverages machine learning and probabilistic modeling techniques to guide synthetic biology in a systematic fashion, without the need for a full mechanistic understanding of the biological system. Using sampling-based optimization, ART provides a set of recommended strains to be built in the next engineering cycle, alongside probabilistic predictions of their production levels. We demonstrate the capabilities of ART on simulated data sets, as well as experimental data from real metabolic engineering projects producing renewable biofuels, hoppy flavored beer without hops, and fatty acids. Finally, we discuss the limitations of this approach, and the practical consequences of the underlying assumptions failing.

Introduction

Metabolic engineering[1] enables us to bioengineer cells to synthesize novel valuable molecules such as renewable biofuels[2,3] or anticancer drugs.[4] The prospects of metabolic engineering to have a positive impact in society are on the rise, as it was considered one of the "Top Ten Emerging Technologies" by the World Economic Forum in 2016.[5] Furthermore, an incoming industrialized biology is expected to improve most human activities: from creating renewable bioproducts and materials, to improving crops and enabling new biomedical applications.[6]

However, the practice of metabolic engineering has been far from systematic, which has significantly hindered its overall impact.[7] Metabolic engineering has remained a collection of useful demonstrations rather than a systematic practice based on generalizable methods. This limitation has resulted in very long development times: for example, it took 150 person-years of effort to produce the antimalarial precursor artemisinin by Amyris; and 575 person-years of effort for Dupont to generate propanediol,[8] which is the base for their commercially available Sorona fabric.[9]

Synthetic biology[10] aims to improve genetic and metabolic engineering by applying systematic engineering principles to achieve a previously specified goal. Synthetic biology encompasses, and goes beyond, metabolic engineering: it also involves non-metabolic tasks such as gene drives able to estinguish malaria-bearing mosquitoes[11] or engineering microbiomes to replace fertilizers.[12] This discipline is enjoying an exponential growth, as it heavily benefits from the byproducts of the genomic revolution: high-throughput multi-omics phenotyping,[13,14] accelerating DNA sequencing[15] and synthesis capabilities,[16] and CRISPR-enabled genetic editing.[17] This exponential growth is reflected in the private investment in the field, which has totalled ∼$12B in the 2009-2018 period and is rapidly accelerating (∼$2B in 2017 to ∼$4B in 2018).[18]

One of the synthetic biology engineering principles used to improve metabolic engineering is the Design-Build-Test-Learn (DBTL[19,20]) cycle—a loop used recursively to obtain a design that satisfies the desired specifications (e.g. a particular titer, rate, yield or prod-

uct). The DBTL cycle's first step is to design (D) a biological system expected to meet the desired outcome. That design is built (B) in the next phase from DNA parts into an appropriate microbial chassis using synthetic biology tools. The next phase involves testing (T) whether the built biological system indeed works as desired in the original design, via a variety of assays: e.g. measurement of production or/and 'omics (transcriptomics, proteomics, metabolomics) data profiling. It is extremely rare that the first design behaves as desired, and further attempts are typically needed to meet the desired specification. The Learn (L) step leverages the data previously generated to inform the next Design step so as to converge to the desired specification faster than through a random search process.

The Learn phase of the DBTL cycle has traditionally been the most weakly supported and developed,[20] despite its critical importance to accelerate the full cycle. The reasons are multiple, although their relative importance is not entirely clear. Arguably, the main drivers of the lack of emphasis on the L phase are: the lack of predictive power for biological systems behavior,[21] the reproducibility problems plaguing biological experiments,[3,22–24] and the traditionally moderate emphasis on mathematical training for synthetic biologists.

Machine learning (ML) arises as an effective tool to predict biological system behavior and empower the Learn phase, enabled by emerging high-throughput phenotyping technologies.[25] Machine learning has been used to produce driverless cars,[26] automate language translation,[27] predict sensitive personal attributes from Facebook profiles,[28] predict pathway dynamics,[29] optimize pathways through translational control,[30] diagnose skin cancer,[31] detect tumors in breast tissues,[32] predict DNA and RNA protein-binding sequences,[33] drug side effects[34] and antibiotic mechanisms of action.[35] However, the practice of machine learning requires statistical and mathematical expertise that is scarce and highly competed for in other fields.[36]

In this paper, we provide a tool that leverages machine learning for synthetic biology's purposes: the Automated Recommendation Tool (ART). ART combines the widely-used and general-purpose open source scikit-learn library[37] with a novel Bayesian[38] ensemble ap-

proach, in a manner that adapts to the particular needs of synthetic biology projects: e.g. low number of training instances, recursive DBTL cycles, and the need for uncertainty quantification. The data sets collected in the synthetic biology field are typically not large enough to allow for the use of deep learning (< 100 instances), but our ensemble model will be able to integrate this approach when high-throughput data generation[14,39] and automated data collection[40] become widely used in the future. ART provides machine learning capabilities in an easy-to-use and intuitive manner, and is able to guide synthetic biology efforts in an effective way.

We showcase the efficacy of ART in guiding synthetic biology by mapping –omics data to production through four different examples: one test case with simulated data and three real cases of metabolic engineering. In all these cases we assume that the -omics data (proteomics in these examples, but it could be any other type: transcriptomics, metabolomics, etc.) can be predictive of the final production (response), and that we have enough control over the system so as to produce any new recommended input. The test case permits us to explore how the algorithm performs when applied to systems that present different levels of difficulty when being "learnt", as well as the effectiveness of using several DTBL cycles. The real metabolic engineering cases involve data sets from published metabolic engineering projects: renewable biofuel production, yeast bioengineering to recreate the flavor of hops in beer, and fatty alcohols synthesis. These projects illustrate what to expect under different typical metabolic engineering situations: high/low coupling of the heterologous pathway to host metabolism, complex/simple pathways, high/low number of conditions, high/low difficulty in learning pathway behavior. We find that ART's ensemble approach can successfully guide the bioengineering process even in the absence of quantitatively accurate predictions. Furthermore, ART's ability to quantify uncertainty is crucial to gauge the reliability of predictions and effectively guide recommendations towards the least known part of the phase space. These experimental metabolic engineering cases also illustrate how applicable the underlying assumptions are, and what happens when they fail.

In sum, ART provides a tool specifically tailored to the synthetic biologist's needs in order to leverage the power of machine learning to enable predictable biology. This combination of synthetic biology with machine learning and automation has the potential to revolutionize bioengineering[25,41,42] by enabling effective inverse design. This paper is written so as to be accessible to both the machine learning and synthetic biology readership, with the intention of providing a much needed bridge between these two very different collectives. Hence, we apologize if we put emphasis on explaining basic machine learning or synthetic biology concepts—they will surely be of use to a part of the readership.

Methods

Key capabilities

ART leverages machine learning to improve the efficacy of bioengineering microbial strains for the production of desired bioproducts (Fig. 1). ART gets trained on available data to produce a model capable of predicting the response variable (e.g. production of the jet fuel limonene) from the input data (e.g. proteomics data, or any other type of data that can be expressed as a vector). Furthermore, ART uses this model to recommend new inputs (e.g. proteomics profiles) that are predicted to reach our desired goal (e.g. improve production). As such, ART bridges the Learn and Design phases of a DBTL cycle.

ART can import data directly from Experimental Data Depot,[43] an online tool where experimental data and metadata are stored in a standardized manner. Alternatively, ART can import EDD-style .csv files, which use the nomenclature and structure of EDD exported files.

By training on the provided data set, ART builds a predictive model for the response as a function of the input variables. Rather than predicting point estimates of the output variable, ART provides the full probability distribution of the predictions. This rigorous quantification of uncertainty enables a principled way to test hypothetical scenarios in-silico, and to guide

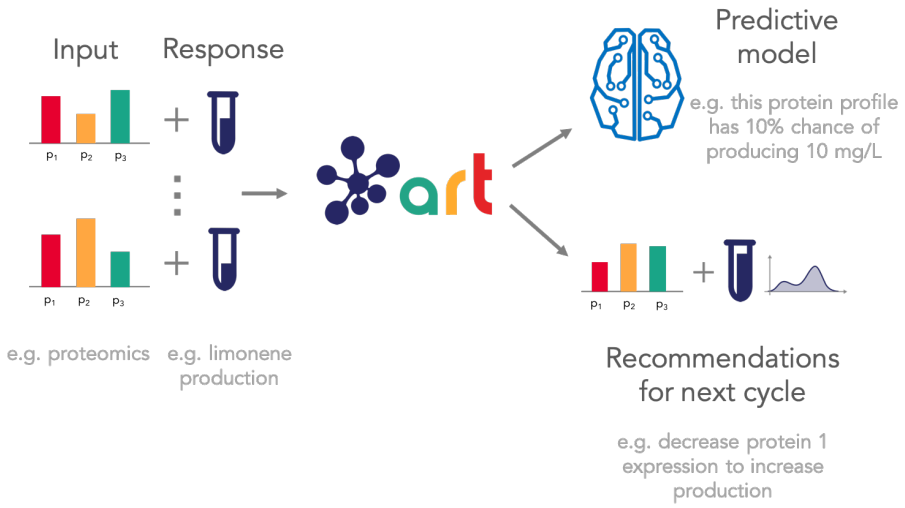


Figure 1: **ART predicts the response from the input and provides recommendations for the next cycle.** ART uses experimental data to i) build a probabilistic predictive model that predicts response (e.g. production) from input variables (e.g. proteomics), and ii) uses this model to provide a set of recommended designs for the next experiment, along with the probabilistic predictions of the response.

design of experiments in the next DBTL cycle. The Bayesian framework chosen to provide the uncertainty quantification is particularly tailored to the type of problems most often encountered in metabolic engineering: sparse data which is expensive and time consuming to generate.

With a predictive model at hand, ART can provide a set of recommendations expected to produce a desired outcome, as well as probabilistic predictions of the associated response. ART supports the following typical metabolic engineering objectives: maximization of the production of a target molecule (e.g. to increase Titer, Rate and Yield, TRY), its minimization (e.g. to decrease the toxicity), as well as specification objectives (e.g. to reach specific level of a target molecule for a desired beer taste profile). Furthermore, ART leverages the probabilistic model to estimate the probability that at least one of the provided recommendations is successful (e.g. it improves the best production obtained so far), and derives how many strain constructions would be required for a reasonable chance to achieve the desired

goal.

While ART can be applied to problems with multiple output variables of interest, it currently supports only the same type of objective for all output variables. Hence, it does not yet support maximization of one target molecule along with minimization of another (see "Success probability calculation" in the supplementary material).

Mathematical methodology

Learning from data: a predictive model through machine learning and a novel Bayesian ensemble approach

By learning the underlying regularities in experimental data, machine learning can provide predictions without a detailed mechanistic understanding (Fig. 2). Training data is used to statistically link an input (i.e. features or independent variables) to an output (i.e. response or dependent variables) through models that are expressive enough to represent almost any relationship. After this training, the models can be used to predict the outputs for inputs that the model has never seen before.

Model selection is a significant challenge in machine learning, since there is a large variety of models available for learning the relationship between response and input, but none of them is optimal for all learning tasks.[44] Furthermore, each model features hyperparameters (i.e. parameters that are set before the training process) that crucially affect the quality of the predictions (e.g. number of trees for random forest or degree of polynomials in polynomial regression), and finding their optimal values is not trivial.

We have sidestepped the challenge of model selection by using an ensemble model approach. This approach takes the input of various different models and has them "vote" for a particular prediction. Each of the ensemble members is trained to perform the same task and their predictions are combined to achieve an improved performance. The examples of the random forest[45] or the super learner algorithm[46] have shown that simple models can be significantly improved by using a set of them (e.g. several types of decision trees in a

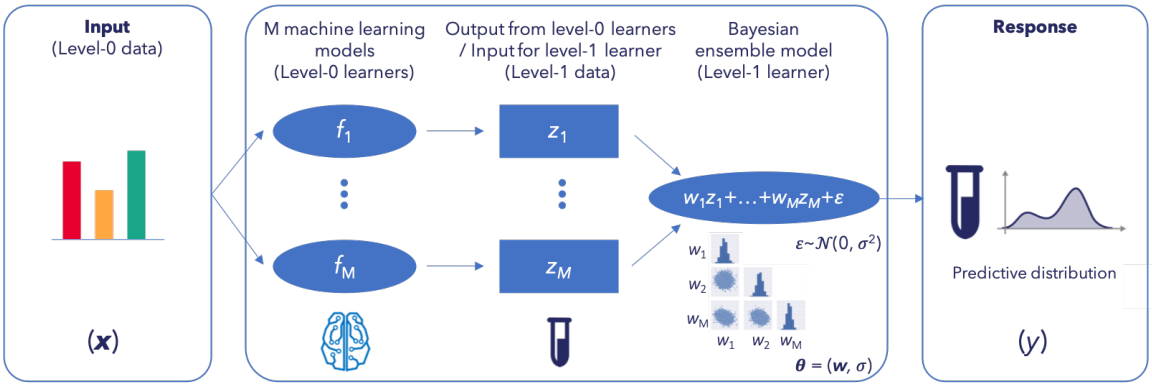


Figure 2: **ART provides a probabilistic predictive model of the response (e.g. production).** ART combines several machine learning models from the scikit-learn library with a novel Bayesian approach to predict the probability distribution of the output. The input to ART is proteomics data (or any other input data in vector format: transcriptomics, gene copy, etc.), which we call level-0 data. This level-0 data is used as input for a variety of machine learning models from the scikit-learn library (level-0 learners) that produce a prediction of production for each model (z_i). These predictions (level-1 data) are used as input for the Bayesian ensemble model (level-1 learner), which weights these predictions differently depending on its ability to predict the training data. The weights w_i and the variance σ are characterized through probability distributions, giving rise to a final prediction in the form of a full probability distribution of response levels.

random forest algorithm). Ensemble model typically either use a set of different models (heterogeneous case) or the same models with different parameters (homogeneous case). We have chosen a heterogeneous ensemble learning approach that uses reasonable hyperparameters for each of the model types, rather than specifically tuning hyperparameters for each of them.

ART uses a novel probabilistic ensemble approach where the weight of each ensemble model is considered a random variable, with a probability distribution inferred by the available data. Unlike other approaches,[47–50] this method does not require the individual models to be probabilistic in nature, hence allowing us to fully exploit the popular scikit-learn library to increase accuracy by leveraging a diverse set of models (see "Related work and novelty of our ensemble approach" in the supplementary material). Our weighted ensemble model approach produces a simple, yet powerful, way to quantify both epistemic and

aleatoric uncertainty—a critical capability when dealing with small data sets and a crucial component of AI in biological research.[51] Here we describe our approach for the single response variable problems, whereas the multiple variables case can be found in the "Multiple response variables" section in the supplementary material. Using a common notation in ensemble modeling we define the following levels of data and learners (see Fig. 2):

- *Level-0 data* ($\mathcal{D}$) represent the historical data consisting of N known instances of inputs and responses, i.e. $\mathcal{D} = \{(\mathbf{x}_n, y_n), n = 1, \ldots, N\}$, where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^D$ is the input comprised of D features and $y \in \mathbb{R}$ is the associated response variable. For the sake of cross-validation, the *level-0 data* are further divided into validation ($\mathcal{D}^{(k)}$) and training sets ($\mathcal{D}^{(-k)}$). $\mathcal{D}^{(k)} \subset \mathcal{D}$ is the kth fold of a K-fold cross-validation obtained by randomly splitting the set $\mathcal{D}$ into K almost equal parts, and $\mathcal{D}^{(-k)} = \mathcal{D} \setminus \mathcal{D}^{(k)}$ is the set $\mathcal{D}$ without the kth fold $\mathcal{D}^{(k)}$. Note that these sets do not overlap and cover the full available data; i.e. $\mathcal{D}^{(k_i)} \cap \mathcal{D}^{(k_j)} = \emptyset, i \neq j$ and $\cup_i \mathcal{D}^{(k_i)} = \mathcal{D}$.

- *Level-0 learners* (f_m) consist of M base learning algorithms $f_m, m = 1, \ldots, M$ used to learn from level-0 training data $\mathcal{D}^{(-k)}$. For ART, we have chosen the following eight algorithms from the scikit-learn library: Random Forest, Neural Network, Support Vector Regressor, Kernel Ridge Regressor, K-NN Regressor, Gaussian Process Regressor, Gradient Boosting Regressor, as well as TPOT (tree-based pipeline optimization tool[52]). TPOT uses genetic algorithms to find the combination of the 11 different regressors and 18 different preprocessing algorithms from scikit-learn that, properly tuned, provides the best achieved cross-validated performance on the training set.

- *Level-1 data* ($\mathfrak{D}_{CV}$) are data derived from $\mathcal{D}$ by leveraging cross-validated predictions of the level-0 learners. More specifically, level-1 data are given by the set $\mathfrak{D}_{CV} = \{(\mathbf{z}_n, y_n), n = 1, \ldots, N\}$, where $\mathbf{z}_n = (z_{1n} \ldots, z_{Mn})$ are predictions for level-0 data ($\mathbf{x}_n \in \mathcal{D}^{(k)}$) of level-0 learners ($f_m^{(-k)}$) trained on observations which are not in fold k ($\mathcal{D}^{(-k)}$), i.e. $z_{mn} = f_m^{(-k)}(\mathbf{x}_n), m = 1, \ldots, M$.

- The *level-1 learner* (F), or metalearner, is a linear weighted combination of level-0 learners, with weights w_m, $m = 1, \ldots, M$ being random variables that are non-negative and normalized to one. Each w_m can be interpreted as the relative confidence in model m. More specifically, given an input $\mathbf{x}$ the response variable y is modeled as:

$$F: \quad y = \boldsymbol{w}^T \mathbf{f}(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2), \tag{1}$$

  where $\boldsymbol{w} = [w_1 \ldots w_M]^T$ is the vector of weights such that $\sum w_m = 1, w_m \geq 0$, $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}) \ldots f_M(\mathbf{x})]^T$ is the vector of level-0 learners, and ε is a normally distributed error variable with a zero mean and standard deviation σ. The constraint $\sum w_m = 1$ (i.e. that the ensemble is a convex combination of the base learners) is empirically motivated but also supported by theoretical considerations.[53] We denote the unknown ensemble model parameters as $\boldsymbol{\theta} \equiv (\boldsymbol{w}, \sigma)$, constituted of the vector of weights and the Gaussian error standard deviation. The parameters $\boldsymbol{\theta}$ are obtained by training F on the level-1 data $\mathfrak{D}_{CV}$ *only*. However, the final model F to be used for generating predictions for new inputs uses these $\boldsymbol{\theta}$, inferred from level-1 data $\mathfrak{D}_{CV}$, and the base learners $f_m, m = 1, \ldots, M$ trained on the full original data set $\mathcal{D}$, rather than only on the level-0 data partitions $\mathcal{D}^{(-k)}$. This follows the usual procedure in developing ensemble learners[54,55] in the context of stacking.[53]

Rather than providing a single point estimate of ensemble model parameters $\boldsymbol{\theta}$ that best fit the training data, a Bayesian model provides a joint probability distribution $p(\boldsymbol{\theta}|\mathcal{D})$ which quantifies the probability that a given set of parameters explains the training data. This Bayesian approach makes it possible to not only make predictions for new inputs but also examine the uncertainty in the model. Model parameters $\boldsymbol{\theta}$ are characterized by full posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ that is inferred from level-1 data. Since this distribution is analytically intractable, we sample from it using the Markov Chain Monte Carlo (MCMC) technique,[56] which samples the parameter space with a frequency proportional to the desired posterior

$p(\boldsymbol{\theta}|\mathcal{D})$ (See "Markov Chain Monte Carlo sampling" section in the supplementary material).

As a result, instead of obtaining a single value as the prediction for the response variable, the ensemble model produces a full distribution that takes into account the uncertainty in model parameters. More precisely, for a new input $\mathbf{x}^*$ (not present in $\mathcal{D}$), the ensemble model F provides the probability that the response is y, when trained with data $\mathcal{D}$ (i.e. the full predictive posterior distribution):

$$p(y|\mathbf{x}^*, \mathcal{D}) = \int p(y|\mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \mathrm{d}\boldsymbol{\theta} = \int \mathcal{N}(y; \boldsymbol{w}^T \mathbf{f}, \sigma^2) p(\boldsymbol{\theta}|\mathcal{D}) \mathrm{d}\boldsymbol{\theta}. \tag{2}$$

where $p(y|\mathbf{x}^*, \boldsymbol{\theta})$ is the predictive distribution of y given input $\mathbf{x}^*$ and model parameters $\boldsymbol{\theta}$, $p(\boldsymbol{\theta}|\mathcal{D})$ is the posterior distribution of model parameters given data $\mathcal{D}$, and $\mathbf{f} \equiv \mathbf{f}(\mathbf{x}^*)$ for the sake of clarity.

Optimization: suggesting next steps

The optimization phase leverages the predictive model described in the previous section to find inputs that are predicted to bring us closer to our objective (i.e. maximize or minimize response, or achieve a desired response level). In mathematical terms, we are looking for a set of N_r suggested inputs $\mathbf{x}_r \in \mathcal{X}; r = 1, \ldots, N_r$, that optimize the response with respect to the desired objective. Specifically, we want a process that:

i) optimizes the predicted levels of the response variable;

ii) can explore the regions of input phase space associated with high uncertainty in predicting response, if desired;

iii) provides a set of different recommendations, rather than only one.

In order to meet these three requirements, we define the optimization problem formally

as

$$\begin{aligned} &\arg\max_{\mathbf{x}} G(\mathbf{x}) \\ &\text{s.t. } \mathbf{x} \in \mathcal{B} \end{aligned}$$

where the surrogate function $G(\mathbf{x})$ is defined as:

$$G(\mathbf{x}) = \begin{cases} (1-\alpha)\mathbb{E}(y) + \alpha \mathrm{Var}(y)^{1/2} & \text{(maximization case)} \\ -(1-\alpha)\mathbb{E}(y) + \alpha \mathrm{Var}(y)^{1/2} & \text{(minimization case)} \\ -(1-\alpha)||\mathbb{E}(y) - y^*||_2^2 + \alpha \mathrm{Var}(y)^{1/2} & \text{(specification case)} \end{cases} \tag{3}$$

depending on which mode ART is operating in (see "Key capabilities" section). Here, y^* is the target value for the response variable, $y = y(\mathbf{x})$, $\mathbb{E}(y)$ and $\mathrm{Var}(y)$ denote the expected value and variance respectively (see "Expected value and variance for ensemble model" in the supplementary material), $||\mathbf{x}||_2^2 = \sum_i x_i^2$ denotes Euclidean distance, and the parameter $\alpha \in [0, 1]$ represents the exploitation-exploration trade-off (see below). The constraint $\mathbf{x} \in \mathcal{B}$ characterizes the lower and upper bounds for each input feature (e.g. protein levels cannot increase beyond a given, physical, limit). These bounds can be provided by the user (see details in the "Implementation" section in the supplementary material); otherwise default values are computed from the input data as described in the "Input space set $\mathcal{B}$" section in the supplementary material.

Requirements i) and ii) are both addressed by borrowing an idea from Bayesian optimization:[57] optimization of a parametrized surrogate function which accounts for both exploitation and exploration. Namely, our objective function $G(\mathbf{x})$ takes the form of the upper confidence bound[58] given in terms of a weighted sum of the expected value and the variance of the response (parametrized by α, Eq. 3). This scheme accounts for both exploitation and exploration: for the maximization case, for example, for $\alpha = 1$ we get $G(\mathbf{x}) = \mathrm{Var}(y)^{1/2}$, so the algorithm suggests next steps that maximize the response variance, thus *exploring* parts

of the phase space where our model shows high predictive uncertainty. For $\alpha = 0$, we get $G(\mathbf{x}) = E(y)$, and the algorithm suggests next steps that maximize the expected response, thus *exploiting* our model to obtain the best response. Intermediate values of α produce a mix of both behaviors. We recommend setting α to values slightly smaller than one for early-stage DBTL cycles, thus allowing for more systematic exploration of the space so as to build a more accurate predictive model in the subsequent DBTL cycles. If the objective is purely to optimize the response, we recommend setting $\alpha = 0$.

In order to address (iii), as well as to avoid entrapment in local optima and search the phase space more effectively, we choose to solve the optimization problem through sampling. More specifically, we draw samples from a target distribution defined as

$$\pi(\mathbf{x}) \propto \exp(G(\mathbf{x}))p(\mathbf{x}), \tag{4}$$

where $p(\mathbf{x}) = \mathcal{U}(\mathcal{B})$ can be interpreted as the uniform 'prior' on the set $\mathcal{B}$, and $\exp(G(\mathbf{x}))$ as the 'likelihood' term of the target distribution. Sampling from π implies optimization of the function G (but not reversely), since the modes of the distribution π correspond to the optima of G. As we did before, we resort to MCMC for sampling. The target distribution is not necessarily differentiable and may well be complex. For example, if it displays more than one mode, as is often the case in practice, there is a risk that a Markov chain gets trapped in one of them. In order to make the chain explore all areas of high probability one can "flatten/melt down" the roughness of the distribution by tempering. For this purpose, we use the Parallel Tempering algorithm[59] for optimization of the objective function through sampling, in which multiple chains at different temperatures are used for exploration of the target distribution (Fig. 3).

Choosing recommendations for the next cycle

After drawing a certain number of samples from $\pi(\mathbf{x})$ we need to choose recommendations for the next cycle, making sure that they are sufficiently different from each other as well

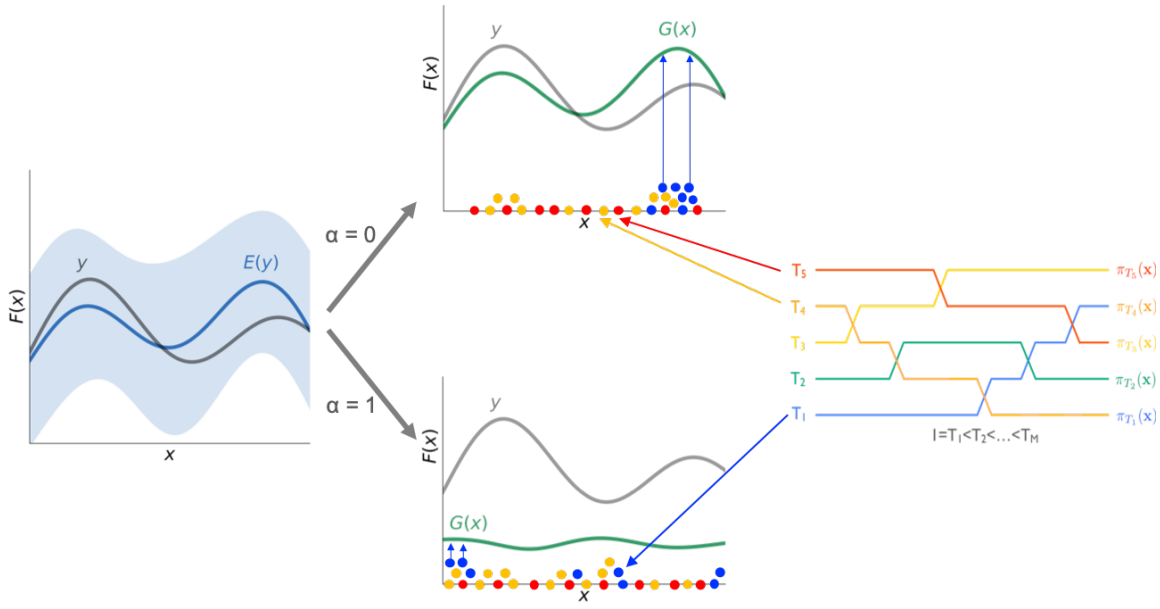


Figure 3: **ART chooses recommendations for next steps by sampling the modes of a surrogate function**. The leftmost panel shows the true response y (e.g. biofuel production to be optimized) as a function of the input $\mathbf{x}$ (e.g. proteomics data), as well as the expected response $E(y)$ after several DBTL cycles, and its 95% confidence interval (blue). Depending on whether we prefer to *explore* the phase space where the model is least accurate or *exploit* the predictive model to obtain the highest possible predicted responses, we will seek to optimize a surrogate function $G(\mathbf{x})$ (Eq. 3), where the exploitation-exploration parameter is $\alpha = 0$ (pure exploitation), $\alpha = 1$ (pure exploration) or anything in between. Parallel-Tempering-based MCMC sampling (center and right side) produces sets of vectors $\mathbf{x}$ (colored dots) for different "temperatures": higher temperatures (red) explore the full phase space, while lower temperature chains (blue) concentrate in the nodes (optima) of $G(\mathbf{x})$. Exchange between different "temperatures" provides more efficient sampling without getting trapped in local optima. Final recommendations (blue arrows) to improve response are provided from the lowest temperature chain, and chosen such that they are not too close to each other and to experimental data (at least 20% difference).

as from the input experimental data. To do so, first we find a sample with optimal $G(\mathbf{x})$ (note that $G(\mathbf{x})$ values are already calculated and stored). We only accept this sample as a recommendation if there is *at least one* feature whose value is different by at least a factor γ (e.g. 20% difference, $\gamma = 0.2$) from the values of that feature in *all* data points $\mathbf{x} \in \mathcal{D}$. Otherwise, we find the next optimal sample and check the same condition. This procedure is repeated until the desired number of recommendations are collected, and the condition

involving γ is satisfied for all previously collected recommendations and all data points. In case all draws are exhausted without collecting the sufficient number of recommendations, we decrease the factor γ and repeat the procedure from the beginning. Pseudo code for this algorithm can be found in Algorithm 1 in the supplementary material. The probability of success for these recommendations is computed as indicated in the "Success probability calculation" section in the supplementary material.

Implementation

ART is implemented Python 3.6 and should be used under this version (see below for software availability). Figure S1 represents the main code structure and its dependencies to external packages. In the "Implementation" section of the supplementary material, we provide an explanation for the main modules and their functions.

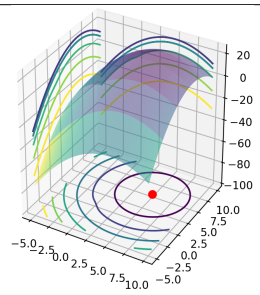
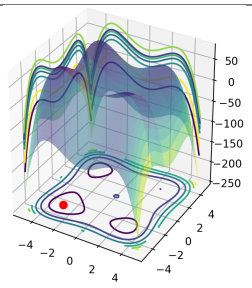
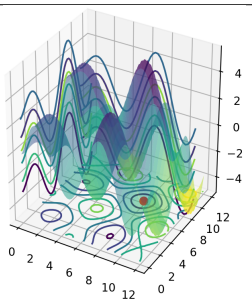
Results and discussion

Using simulated data to test ART

Synthetic data sets allow us to test how ART performs when confronted by problems of different difficulty and dimensionality, as well as gauge the effect of the availability of more training data. In this case, we tested the performance of ART for 1–10 DBTL cycles, three problems of increasing difficulty (F_E, F_M and F_D, see Table 1), and three different dimensions of input space ($D = 2, 10$ and 50, Fig. 4). We simulated the DBTL processes by starting with a training set given by 16 strains (Latin Hypercube[60] draws) and the associated measurements (from Table 1 functions). We limited ourselves to the maximization case, and at each DBTL cycle, generated 16 recommendations that maximize the objective function given by Eq. (3). This choice mimicked triplicate experiments in the 48 wells of throughput of a typical automated fermentation platform.[61] We employed a *tempering* strategy for the exploitation-exploration parameter, i.e. assign $\alpha = 0.9$ at start for an exploratory optimiza-

tion, and gradually decreased the value to $\alpha = 0$ in the final DBTL cycle for the exploitative maximization of the production levels.

Table 1: Functions presenting different levels of difficulty to being learnt, used to produce synthetic data and test ART's performance (Fig. 4).

Easy	$F_E(\mathbf{x}) = -\frac{1}{d}\sum_i^d (x_i - 5)^2 + \exp\left(-\sum_i x_i^2\right) + 25$	
Medium	$F_M(\mathbf{x}) = \frac{1}{d}\sum_i^d \left(x_i^4 - 16x_i^2 + 5x_i\right)$	
Difficult	$F_D(\mathbf{x}) = \sum_i^d \sqrt{x_i}\sin(x_i)$	

ART performance improves significantly as more data are accrued with additional DTBL cycles. Whereas the prediction error, given in terms of Mean Average Error (MAE), remains constantly low for the training set (i.e. ART is always able to reliably predict data it has already seen), the MAE for the test data (data ART has not seen) in general decreases markedly only with the addition of more DBTL cycles (Fig. S2). The exceptions are the most complicated problems: those exhibiting highest dimensionality ($D = 50$), where MAE stays approximately constant, and the difficult function F_D, which exhibits a slower decrease.

Furthermore, the best production among the 16 recommendations obtained in the simulated process increases monotonically with more DBTL cycles: faster for easier problems and lower dimensions and more slowly for harder problems and higher dimensions. Finally, the uncertainty in those predictions decreases as more DBTL cycles proceed (Fig. 4). Hence, more data (DBTL cycles) almost always translates into better predictions and production. However, we see that these benefits are rarely reaped with only the 2 DBTL cycles customarily used in metabolic engineering (see examples in the next sections): ART (and ML in general) becomes only truly efficient when using 5–10 DBTL cycles.

Different experimental problems involve different levels of difficulty when being learnt (i.e. being predicted accurately), and this can only be assessed empirically. Low dimensional problems can be easily learnt, whereas exploring and learning a 50-dimensional landscape is very slow (Fig. 4). Difficult problems (i.e. less monotonic landscapes) take more data to learn and traverse than easier ones. We will showcase this point in terms of real experimental data when comparing the biofuel project (easy) versus the dodecanol project (hard) below. However, it is (difficult) to decide a priori whether a given real data project or problem will be easy or hard to learn—the only way to determine this is by checking the improvements in prediction accuracy as more data is added. In any case, a starting point of at least ~ 100 instances is highly recommendable to obtain proper statistics.

Improving the production of renewable biofuel

The optimization of the production of the renewable biofuel limonene through synthetic biology will be our first demonstration of ART using real-life experimental data. Renewable biofuels are almost carbon neutral because they only release into the atmosphere the carbon dioxide that was taken up in growing the plant biomass they are produced from. Biofuels from renewable biomass have been estimated to be able to displace ∼30% of petroleum consumption[62] and are seen as the most viable option for decarbonizing sectors that are challenging to electrify, such as heavy-duty freight and aviation.[63]

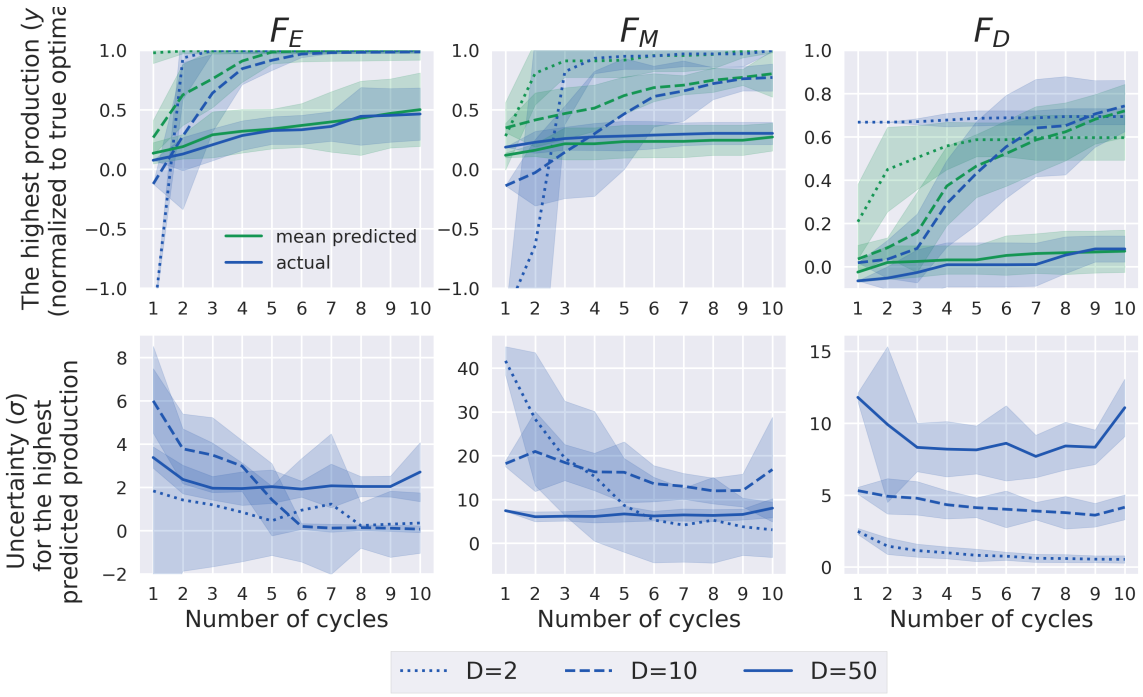


Figure 4: **ART performance improves significantly by proceeding beyond the usual two Design-Build-Test-Learn cycles.** Here we show the results of testing ART's performance with synthetic data obtained from functions of different levels of complexity (see Table 1), different phase space dimensions (2, 10 and 50), and different amounts of training data (DBTL cycles). The top row presents the results of the simulated metabolic engineering in terms of highest production achieved so far for each cycle (as well as the corresponding ART predictions). The production increases monotonically with a rate that decreases as the problem is harder to learn, and the dimensionality increases. The bottom row shows the uncertainty in ART's production prediction, given by the standard deviation of the response distribution (Eq. 2). This uncertainty decreases markedly with the number of DBTL cycles, except for the highest number of dimensions. In each plot, lines and shaded areas represent the estimated mean values and 95% confidence intervals, respectively, over 10 repeated runs. Mean Absolute Error (MAE) and training and test set definitions can be found in Fig. S2.

Limonene is a molecule that can be chemically converted to several pharmaceutical and commodity chemicals.[64] If hydrogenated, for example, it has low freezing point and is immiscible with water, characteristics which are ideal for next generation jet-biofuels and fuel additives that enhance cold weather performance.[65,66] Limonene has been traditionally obtained from plant biomass, as a byproduct of orange juice production, but fluctuations in availability, scale and cost limit its use as biofuel.[67] The insertion of the plant genes re-

sponsible for the synthesis of limonene in a host organism (e.g. a bacteria), however, offers a scalable and cheaper alternative through synthetic biology. Limonene has been produced in *E. coli* through an expansion of the celebrated mevalonate pathway (Fig. 1a in Alonso-Gutierrez et al.[68]), used to produce the antimalarial precursor artemisinin[69] and the biofuel farnesene,[70] and which forms the technological base on which the company Amyris was founded (valued ∼$300M ca. 2019). This version of the mevalonate pathway is composed of seven genes obtained from such different organisms as *S. cerevesiae*, *S. aureus*, and *E. coli*, to which two genes have been added: a geranyl-diphosphate synthase and a limonene synthase obtained from the plants *A. grandis* and *M. spicata*, respectively.

For this demonstration, we use historical data from Alonso-Gutierrez et al.[71], where 27 different variants of the pathway (using different promoters, induction times and induction strengths) were built. Data collected for each variant involved limonene production and protein expression for each of the nine proteins involved in the synthetic pathway. These data were used to feed Principal Component Analysis of Proteomics (PCAP),[71] an algorithm using principal component analysis to suggest new pathway designs. The PCAP recommendations, used to engineer new strains, resulted in a 40% increase in production for limonene, and 200% for bisabolene (a molecule obtained from the same base pathway). This small amount of available instances (27) to train the algorithms is typical of synthetic biology/metabolic engineering projects. Although we expect automation to change the picture in the future,[25] the lack of large amounts of data has determined our machine learning approach in ART (i.e. no deep neural networks).

ART is able to not only recapitulate the successful predictions obtained by PCAP improving limonene production, but also provides a systematic way to obtain them as well as the corresponding uncertainty. In this case, the training data for ART are the concentrations for each of the nine proteins in the heterologous pathway (input), and the production of limonene (response). The objective is to maximize limonene production. We have data for two DBTL cycles, and we use ART to explore what would have happened if we have used

ART instead of PCAP for this project.

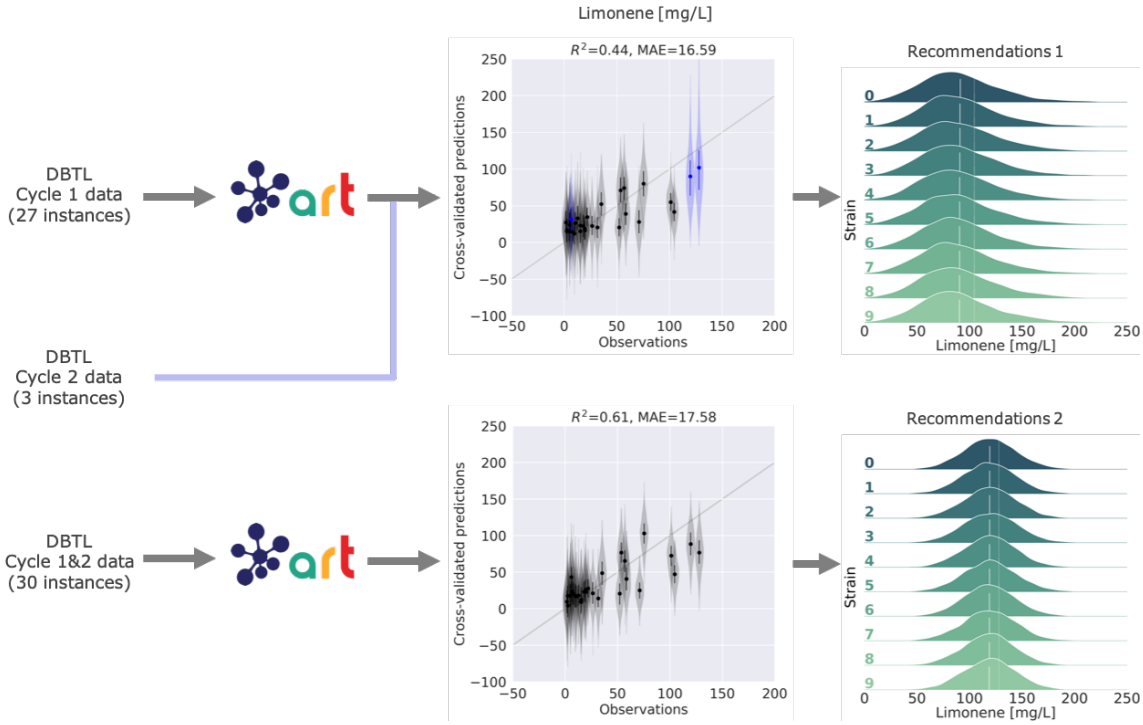


Figure 5: **ART provides effective recommendations to improve renewable biofuel (limonene) production**. We used the first DBTL cycle data (27 strains, top) to train ART and recommend new protein targets (top right). The ART recommendations were very similar to the protein profiles that eventually led to a 40% increase in production (Fig. 6). ART predicts mean production levels for the second DBTL cycle strains which are very close to the experimentally measured values (three blue points in top graph). Adding those three points from DBTL cycle 2 provides a total of 30 strains for training that lead to recommendations predicted to exhibit higher production and narrower distributions (bottom right). Uncertainty for predictions is shown as probability distributions for recommendations and violin plots for the cross-validated predictions. R^2 and Mean Absolute Error (MAE) values are only for cross-validated mean predictions (black data points).

We used the data from DBLT cycle 1 to train ART and recommend new strain designs (i.e. protein profiles for the pathway genes, Fig. 5). The model trained with the initial 27 instances provided reasonable cross-validated predictions for production of this set ($R^2 = 0.44$), as well as the three strains which were created for DBTL cycle 2 at the behest of PCAP (Fig. 5). This suggests that ART would have easily recapitulated the PCAP results. Indeed, the ART

recommendations are very close to the PCAP recommendations (Fig. 6). Interestingly, we see that while the quantitative predictions of each of the individual models were not very accurate, they all signaled towards the same direction in order to improve production, hence showing the importance of the ensemble approach (Fig. 6).

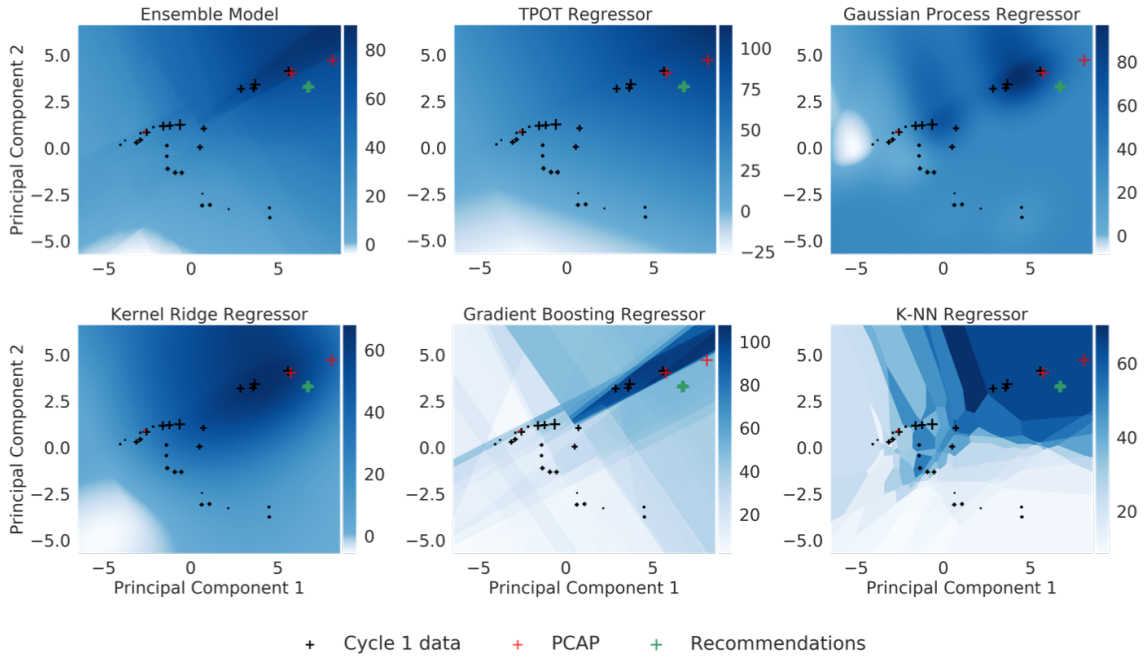


Figure 6: **All machine learning algorithms point in the same direction to improve limonene production, in spite of quantitative differences in prediction**. Cross sizes indicate experimentally measured limonene production in the proteomics phase space (first two principal components shown from principal component analysis, PCA). The color heatmap indicates the limonene production predicted by a set of base regressors and the final ensemble model (top left) that leverages all the models and conforms the base algorithm used by ART. Although the models differ significantly in the actual quantitative predictions of production, the same qualitative trends can be seen in all models (i.e. explore upper right quadrant for higher production), justifying the ensemble approach used by ART. The ART recommendations (green) are very close to the PCAP recommendations (red) that were experimentally tested to improve production by 40%.

Training ART with experimental results from DBTL cycles 1 and 2 results in even better predictions ($R^2 = 0.61$), highlighting the importance of the availability of large amounts of data to train ML models. This new model suggests new sets of strains predicted to produce

even higher amounts of limonene. Importantly, the uncertainty in predicted production levels is significantly reduced with the additional data points from cycle 2.

Brewing hoppy beer without hops by bioengineering yeast

Our second example involves bioengineering yeast (*S. cerevisiae*) to produce hoppy beer without the need for hops.[72] To this end, the ethanol-producing yeast used to brew the beer, was modified to also synthesize the metabolites linalool (L) and geraniol (G), which impart hoppy flavor (Fig. 2B in Denby et al.[72]). Synthesizing linalool and geraniol through synthetic biology is economically advantageous because growing hops is water and energetically intensive, and their taste is highly variable from crop to crop. Indeed, a startup (Berkeley Brewing Science[73]) was generated from this technology.

ART is able to efficiently provide the proteins-to-production mapping that required three different types of mathematical models in the original publication, paving the way for a systematic approach to beer flavor design. The challenge is different in this case as compared to the previous example (limonene): instead of trying to maximize production, the goal is to reach a particular level of linalool and geraniol so as to match a known beer tasting profile (e.g. Pale Ale, Torpedo or Hop Hunter, Fig. 7). ART can provide this type of recommendations, as well. For this case, the inputs are the expression levels for the four different proteins involved in the pathway, and the response are the concentrations of the two target molecules (L and G), for which we have desired targets. We have data for two DBTL cycles involving 50 different strains/instances (19 instances for the first DBTL cycle and 31 for the second one, Fig. 7). As in the previous case, we use this data to simulate the outcomes we would have obtained in case ART had been available for this project.

The first DBTL cycle provides a very limited number of 19 instances to train ART, which performs passably on this training set, and poorly on the test set provided by the 31 instances from DBTL cycle 2 (Fig. 7). Despite this small amount of training data, the model trained in DBTL cycle 1 is able to recommend new protein profiles that are predicted to reach the

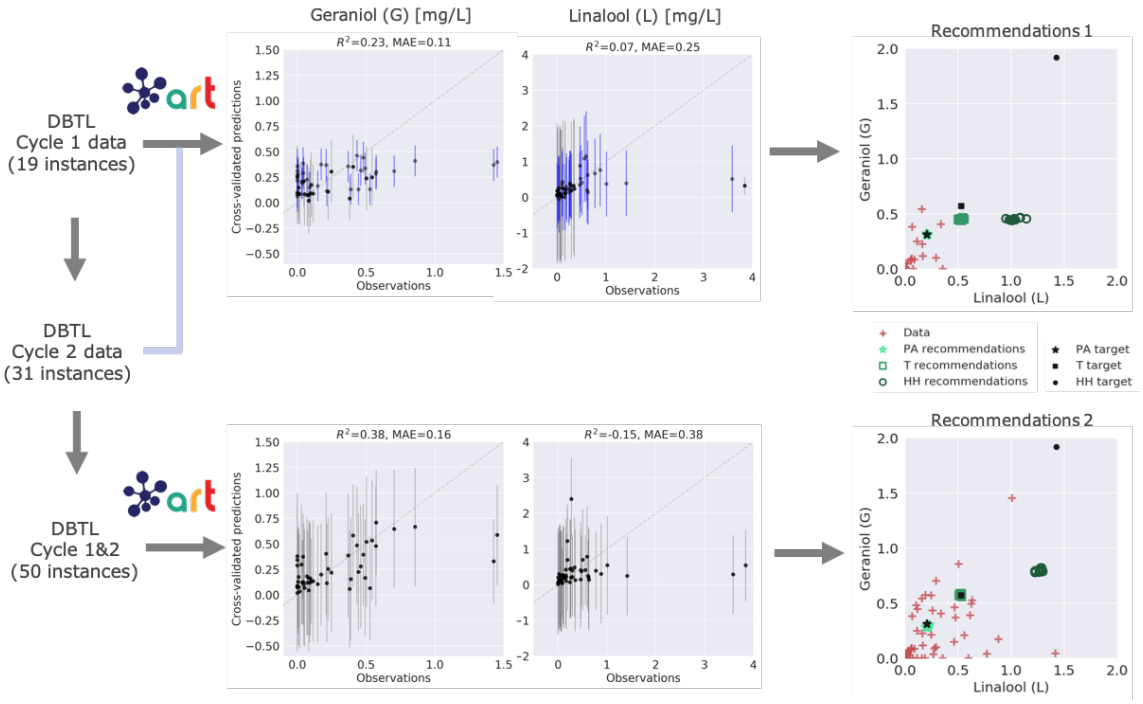


Figure 7: **ART produces effective recommendations to bioengineer yeast to produce hoppy beer without hops.** The 19 instances in the first DBTL cycle were used to train ART, but it did not show an impressive predictive power (particularly for L, top middle). In spite of it, ART is still able to recommend protein profiles predicted to reach the Pale Ale (PA) target flavor profile, and others which were close to the Torpedo (T) metabolite profile (top right, green points showing mean predictions). Adding the 31 strains for the second DBTL cycle improves predictions for G but not for L (bottom). The expanded range of values for G & L provided by cycle 2 allows ART to recommend profiles which are predicted to reach targets for both beers (bottom right), but not Hop Hunter (HH). Hop Hunter displays a very different metabolite profile from the other beers, well beyond the range of experimentally explored values of G & L, making it impossible for ART to extrapolate that far. Notice that none of the experimental data (red crosses) matched exactly the desired targets (black symbols), but the closest ones were considered acceptable. R^2 and Mean Absolute Error (MAE) values are for cross-validated mean predictions (black data points) only. Bars indicate 95% credible interval of the predictive posterior distribution.

Pale Ale target (Fig. 7). Similarly, this DBTL cycle 1 model was almost able to reach (in predictions) the L and G levels for the Torpedo beer, which will be finally achieved in DBTL cycle 2 recommendations, once more training data is available. For the Hop Hunter beer, recommendations from this model were not close to the target.

The model for the second DBTL cycle leverages the full 50 instances from cycles 1 and 2 for training and is able to provide recommendations predicted to attain two out of three targets. The Pale Ale target L and G levels were already predicted to be matched in the first cycle; the new recommendations are able to maintain this beer profile. The Torpedo target was almost achieved in the first cycle, and is predicted to be reached in the second cycle recommendations. Finally, Hop Hunter target L and G levels are very different from the other beers and cycle 1 results, so neither cycle 1 or 2 recommendations can predict protein inputs achieving this taste profile. ART has only seen two instances of high levels of L and G and cannot extrapolate well into that part of the metabolic phase space. ART's exploration mode, however, can suggest experiments to explore this space.

Quantifying the prediction uncertainty is of fundamental importance to gauge the reliability of the recommendations, and the full process through several DBTL cycles. In the end, the fact that ART was able to recommend protein profiles predicted to match the Pale Ale and Torpedo taste profiles only indicates that the optimization step (see "Optimization: suggesting next steps" section) works well. The actual recommendations, however, are only as good as the predictive model. In this regard, the predictions for L and G levels shown in Fig. 7 (right side) may seem deceptively accurate, since they are only showing the average predicted production. Examining the full probability distribution provided by ART shows a very broad spread for the L and G predictions (much broader for L than G, Fig. S3). These broad spreads indicate that the model still has not converged and that recommendations will probably change significantly with new data. Indeed, the protein profile recommendations for the Pale Ale changed markedly from DBTL cycle 1 to 2, although the average metabolite predictions did not (left panel of Fig. S4). All in all, these considerations indicate that quantifying the uncertainty of the predictions is important to foresee the smoothness of the optimization process.

At any rate, despite the limited predictive power afforded by the cycle 1 data, ART recommendations guide metabolic engineering effectively. For both of the Pale Ale and

Torpedo cases, ART recommends exploring parts of the proteomics phase space such that the final protein profiles (that were deemed close enough to the targets), lie between the first cycle data and these recommendations (Fig. S4). Finding the final target becomes, then, an interpolation problem, which is much easier to solve than an extrapolation one. These recommendations improve as ART becomes more accurate with more DBTL cycles.

Improving dodecanol production

The final example is one of a failure (or at least a mitigated success), from which as much can be learnt as from the previous successes. Opgenorth et al.[74] used machine learning to drive two DBTL cycles to improve production of 1-dodecanol in *E. coli*, a medium-chain fatty acid used in detergents, emulsifiers, lubricants and cosmetics. This example illustrates the case in which the assumptions underlying this metabolic engineering and modeling approach (mapping proteomics data to production) fail. Although a ∼20% production increase was achieved, the machine learning algorithms were not able to produce accurate predictions with the low amount of data available for training, and the tools available to reach the desired target protein levels were not accurate enough.

This project consisted of two DBTL cycles comprising 33 and 21 strains, respectively, for three alternative pathway designs (Fig. 1 in Opgenorth et al.[74], Table S4). The use of replicates increased the number of instances available for training to 116 and 69 for cycle 1 and 2, respectively. The goal was to modulate the protein expression by choosing Ribosome Binding Sites (RBSs, the mRNA sites to which ribosomes bind in order to translate proteins) of different strengths for each of the three pathways. The idea was for the machine learning to operate on a small number of variables (∼3 RBSs) that, at the same time, provided significant control over the pathway. As in previous cases, we will show how ART could have been used in this project. The input for ART in this case consists of the concentrations for each of three proteins (different for each of the three pathways), and the goal was to maximize 1-dodecanol production.

The first challenge involved the limited predictive power of machine learning for this case. This limitation is shown by ART's completely compromised prediction accuracy (Fig. 8). The causes seem to be twofold: a small training set and a strong connection of the pathway to the rest of host metabolism. The initial 33 strains (116 instances) were divided into three different designs (Table S4), decimating the predictive power of ART (Figs. 8, S5 and S6). Now, it is complicated to estimate the number of strains needed for accurate predictions because that depends on the complexity of the problem to be learnt (see "Using simulated data to test ART" section). In this case, the problem is harder to learn than the previous two examples: the mevalonate pathway used in those examples is fully exogenous (i.e. built from external genetic parts) to the final yeast host and hence, free of the metabolic regulation that is certainly present for the dodecanol producing pathway. The dodecanol pathway depends on fatty acid biosynthesis which is vital for cell survival (it produces the cell membrane), and has to be therefore tightly regulated.[75] This characteristic makes it more difficult to learn its behavior by ART using only dodecanol synthesis pathway protein levels (instead of adding also proteins from other parts of host metabolism).

A second challenge, compounding the first one, involves the inability to reach the target protein levels recommended by ART to increase production. This difficulty precludes not only bioengineering, but also testing the validity of the ART model. For this project, both the mechanistic (RBS calculator[76,77]) and machine learning-based (EMOPEC[78]) tools proved to be very inaccurate for bioengineering purposes: e.g. a prescribed 6-fold increase in protein expression could only be matched with a 2-fold increase. Moreover, non-target effects (i.e. changing the RBS for a gene significantly affects protein expression for other genes in the pathway) were abundant, further adding to the difficulty. While unrelated directly to ART performance, these effects highlight the importance of having enough control over ART's input (proteins in this case) to obtain satisfactory bioengineering results.

A third, unexpected, challenge was the inability of constructing several strains in the Build phase due to toxic effects engendered by the proposed protein profiles (Table S4).

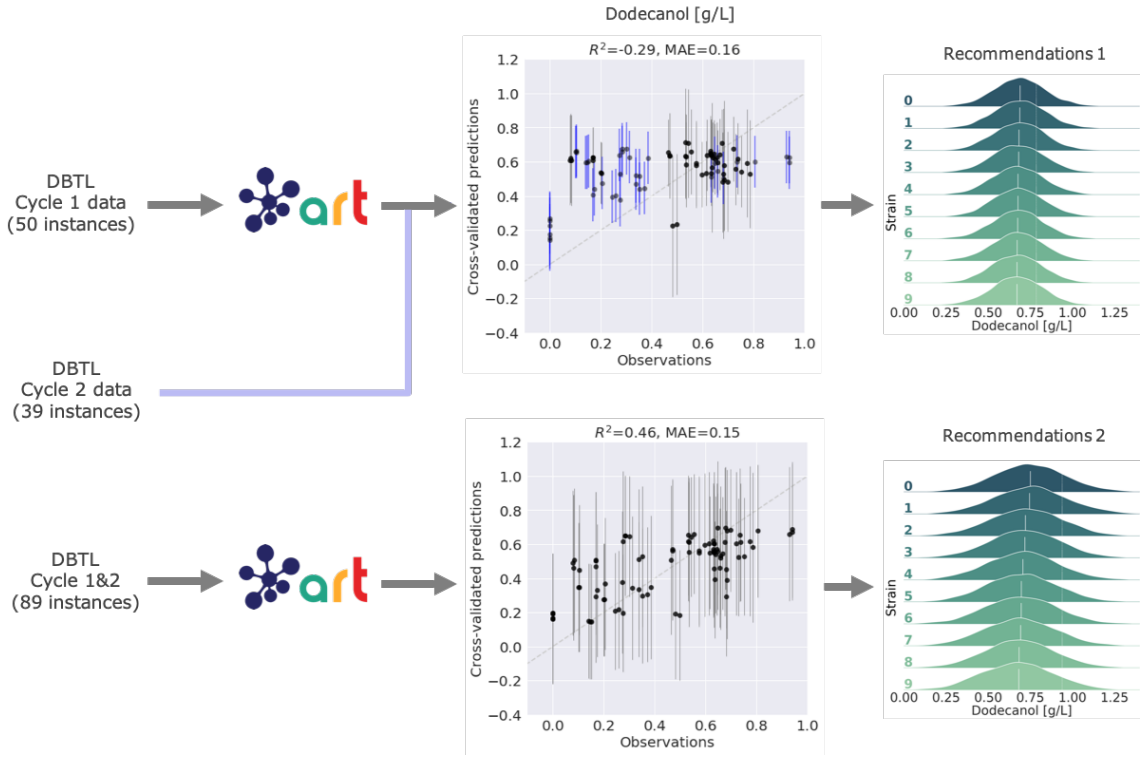


Figure 8: **ART's predictive power is heavily compromised in the dodecanol production example**. Although the 50 instances available for cycle 1 (top) almost double the 27 available instances for the limonene case (Fig. 5), the predictive power of ART is heavily compromised ($R^2 = -0.29$ for cross-validation) by the strong tie of the pathway to host metabolism (fatty acid production), and the scarcity of data. The poor predictions for the test data from cycle 2 (in blue) confirm the lack of predictive power. Adding data from both cycles (1 and 2) improves predictions notably (bottom). These data and model refer to the first pathway in Fig. 1B from Opgenorth et al.[74]. The cases for the other two pathways produce similar conclusions (Figs. S5 and S6). R^2 and Mean Absolute Error (MAE) values are only for cross-validated mean predictions (black data points). Bars indicate 95% credible interval of the predictive posterior distribution.

This phenomenon materialized through mutations in the final plasmid in the production strain or no colonies after the transformation. The prediction of these effects in the Build phase represents an important target for future ML efforts, in which tools like ART can have an important role. A better understanding of this phenomenon may not only enhance bioengineering but also reveal new fundamental biological knowledge.

These challenges highlight the importance of carefully considering the full experimental

design before leveraging machine learning to guide metabolic engineering.

Conclusion

ART is a tool that not only provides synthetic biologists easy access to machine learning techniques, but can also systematically guide bioengineering and quantify uncertainty. ART takes as input a set of vectors of measurements (e.g. a set of proteomics measurements for several proteins, or transcripts for several genes) along with their corresponding systems responses (e.g. associated biofuel production) and provides a predictive model, as well as recommendations for the next round (e.g. new proteomics targets predicted to improve production in the next round).

ART combines the methods from the scikit-learn library with a novel Bayesian ensemble approach and MCMC sampling, and is optimized for the conditions encountered in metabolic engineering: small sample sizes, recursive DBTL cycles and the need for uncertainty quantification. ART's approach involves an ensemble where the weight of each model is considered a random variable with a probability distribution inferred from the available data. Unlike other approaches, this method does not require the ensemble models to be probabilistic in nature, hence allowing us to fully exploit the popular scikit-learn library to increase accuracy by leveraging a diverse set of models. This weighted ensemble model produces a simple, yet powerful, approach to quantify uncertainty (Fig. 5), a critical capability when dealing with small data sets and a crucial component of AI in biological research.[51] While ART is adapted to synthetic biology's special needs and characteristics, its implementation is general enough that it is easily applicable to other problems of similar characteristics. ART is perfectly integrated with the Experiment Data Depot[43] and the Inventory of Composable Elements,[79] forming part of a growing family of tools that standardize and democratize synthetic biology.

We have showcased the use of ART in a case with synthetic data sets and three real metabolic engineering cases from the published literature. The synthetic data case involves data generated for several production landscapes of increasing complexity and dimensionality.

This case allowed us to test ART for different levels of difficulty of the production landscape to be learnt by the algorithms, as well as different numbers of DBTL cycles. We have seen that while easy landscapes provide production increases readily after the first cycle, more complicated ones require >5 cycles to start producing satisfactory results (Fig. 4). In all cases, results improved with the number of DBTL cycles, underlying the importance of designing experiments that continue for ∼10 cycles rather than halting the project if results do not improve in the first few cycles.

The demonstration cases using real data involve engineering *E. coli* and *S. cerevisiae* to produce the renewable biofuel limonene, synthesize metabolites that produce hoppy flavor in beer, and generate dodecanol from fatty acid biosynthesis. Although we were able to produce useful recommendations with as low as 27 (limonene, Fig. 5) or 19 (hopless beer, Fig. 7) instances, we also found situations in which larger amounts of data (50 instances) were insufficient for meaningful predictions (dodecanol, Fig. 8). It is impossible to determine a priori how much data will be necessary for accurate predictions, since this depends on the difficulty of the relationships to be learnt (e.g. the amount of coupling between the studied pathway and host metabolism). However, one thing is clear—two DBTL cycles (which was as much as was available for all these examples) are rarely sufficient for guaranteed convergence of the learning process. We do find, though, that accurate quantitative predictions are not required to effectively guide bioengineering—our ensemble approach can successfully leverage qualitative agreement between the models in the ensemble to compensate for the lack of accuracy (Fig. 6). Uncertainty quantification is critical to gauge the reliability of the predictions (Fig. 5), anticipate the smoothness of the recommendation process through several DBTL cycles (Figs S3 and S4), and effectively guide the recommendations towards the least understood part of the phase space (exploration case, Fig. 3). We have also explored several ways in which the current approach (mapping –omics data to production) can fail when the underlying asssumptions break down. Among the possible pitfalls is the possibility that recommended target protein profiles cannot be accurately reached, since the tools to

produce specified protein levels are still imperfect; or because of biophysical, toxicity or regulatory reasons. These areas need further investment in order to accelerate bioengineering and make it more reliable, hence enabling design to a desired specification.

While ART is a useful tool in guiding bioengineering, it represents just an initial step in applying machine learning to synthetic biology. Future improvements under consideration include adding a pathway cost ($) function, the inclusion of classification problems, adding new optimization methods (e.g. to include the case of discrete input variables), incorporating covariance of level-0 models into the ensemble model, and incorporating input space errors into learners. These may not be the preferred list of improvements for every user, so ART's dual license allows for modification by third parties for research purposes, as long as the modifications are offered to the original repository. Hence, users are encouraged to enhance it in ways that satisfy their needs. A commercial use license is also available (see below for details).

ART provides effective decision-making in the context of synthetic biology and facilitates the combination of machine learning and automation that might disrupt synthetic biology.[25] Combining ML with recent developments in macroscale lab automation,[61,80] microfluidics[21,40,81–83] and cloud labs[84] may enable self-driving laboratories,[41,42] which augment automated experimentation platforms with artificial intelligence to facilitate autonomous experimentation. We believe that fully leveraging AI and automation can catalyze a similar step forward in synthetic biology as CRISPR-enabled genetic editing, high-throughput multi-omics phenotyping, and exponentially growing DNA synthesis capabilities have produced in the recent past.

Competing interests

The authors declare that they have no competing interests.

Author's contributions

Z.C., T.R. and H.G.M. conceived the original idea. T.R. and Z.C. developed methodology, designed the software, wrote the code and performed computer experiments. T.R. designed simulated benchmarks and performed numerical experiments. T.R. analyzed all results. K.W. wrote tests and documented the code. H.G.M. and T.R. wrote the paper.

Data availability

The experimental data analyzed in this paper can be found both in the Experiment Data Depot,[43] in the following studies:

Study	Link
Biofuel	`https://public-edd.jbei.org/s/pcap/`
Hopless beer	`https://public-edd.agilebiofoundry.org/s/hopless-beer/`
	`https://public-edd.agilebiofoundry.org/s/hopless-beer-cycle-2/`
Dodecanol	`https://public-edd.jbei.org/s/ajinomoto/`

and as .csv files in the Github repository (see below).

Software availability

ART's dual license allows for free non-commercial use for academic institutions. Modifications should be fed back to the original repository for the benefit of all users. A separate commercial use license is available from Berkeley Lab (ipo@lbl.gov). See `https://github.com/JBEI/ART` for software and licensing details.

Acknowledgement

This work was part of the DOE Joint BioEnergy Institute (`http://www.jbei.org`) and part of the Agile BioFoundry (`http://agilebiofoundry.org`) supported by the U. S. Depart-

ment of Energy, Energy Efficiency and Renewable Energy, Bioenergy Technologies Office, Office of Science, through contract DE-AC02-05CH11231 between Lawrence Berkeley National Laboratory and the U. S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (`http://energy.gov/downloads/doe-public-access-plan`). This research is also supported by the Basque Government through the BERC 2014-2017 program and by Spanish Ministry of Economy and Competitiveness MINECO: BCAM Severo Ochoa excellence accreditation SEV-2013-0323.

We acknowledge and thank Peter St. John, Joonhoon Kim, Amin Zargar, Henrique De Paoli, Sean Peisert and Nathan Hillson for reviewing the manuscript and for their helpful suggestions.

Supplementary material

Mathematical Methods

Markov Chain Monte Carlo sampling

The posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ (probability that the parameters $\boldsymbol{\theta}$ fit the data $\mathcal{D}$, used in Eq. 2) is obtained by applying Bayes' formula, i.e. it is defined through a prior $p(\boldsymbol{\theta})$ and a likelihood function $p(\mathcal{D}|\boldsymbol{\theta})$ as

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}).$$

We define the prior to be $p(\boldsymbol{\theta}) = p(\boldsymbol{w})p(\sigma)$, where $p(\boldsymbol{w})$ is a Dirichlet distribution with uniform parameters, which ensures the constraint on weights (i.e. they all add to one) is satisfied, and $p(\sigma)$ is a half normal distribution with mean and standard deviation set to 0 and 10, respectively. The likelihood function follows directly from Eq. (1) as

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \boldsymbol{\theta}), \quad p(y_n|\mathbf{x}_n, \boldsymbol{\theta}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(y_n - \boldsymbol{w}^T\mathbf{f}(\mathbf{x}_n))^2}{2\sigma^2}\right\}.$$

Expected value and variance for ensemble model

From Eq. (1) , we can easily compute the expected value

$$\mathbb{E}(y) = \mathbb{E}(\boldsymbol{w}^T\mathbf{f} + \varepsilon) = \mathbb{E}(\boldsymbol{w})^T\mathbf{f} \tag{5}$$

and variance

$$\mathrm{Var}(y) = \mathbf{f}^T\mathrm{Var}(\boldsymbol{w})\mathbf{f} + \mathrm{Var}(\varepsilon) \tag{6}$$

of the response, which will be needed for are used in the optimization phase in order to create the surrogate function $G(\mathbf{x})$ (Eq. 3). The expected value and variance of $\boldsymbol{w}$ and ε are estimated through sample mean and variance using samples from the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$.

Please note that although we have modeled $p(y|\mathbf{x}^*, \boldsymbol{\theta})$ to be Gaussian (Eq. 1), the predictive posterior distribution $p(y|\mathbf{x}^*, \mathcal{D})$ (Eq. 2) is not Gaussian due to the complexity of $p(\boldsymbol{\theta}|\mathcal{D})$ arising from the data and other constraints.

It is important to note that our modeling approach provides quantification of both epistemic and aleatoric uncertainty, through the first and second terms in Eq. (6), respectively. Epistemic (systematic) uncertainty accounts for uncertainty in the model and aleatoric (statistical) uncertainty describes the amount of noise inherent in the data.[85] While epistemic uncertainty can be eventually explained away given enough data, we need to model it accurately in order to properly capture situations not encountered in the training set. Modeling epistemic uncertainty is therefore important for small data problems, while aleatoric uncertainty is more relevant for large data problems. In general, it is useful to characterize the uncertainties within a model, as this enables understanding which uncertainties have the potential of being reduced.[86]

Related work and novelty of our ensemble approach

Our ensemble approach is based on stacking[53]—a method where different ensemble members are trained on the same training set and whose outputs are then combined, as opposed to techniques that manipulate the training set (e.g. bagging[87]) or those that sequentially add new models into the ensemble (e.g. boosting[88]). Different approaches for constructing ensemble of models using the Bayesian framework have been already considered. For example, Bayesian Model Averaging (BMA)[47] builds an ensemble model as a linear combination of the individual members in which the weights are given by the posterior probabilities of models. The weights therefore crucially depend on marginal likelihood under each model, which is challenging to compute. BMA accounts for uncertainty about which model is correct but assumes that only one of them is, and as a consequence, it has the tendency of selecting the one model that is the closest to the generating distribution. Agnostic Bayesian learning of ensembles[89] differs from BMA in the way the weights are calculated. Instead of finding the

best predictor from the model class (assuming that the observed data is generated by one of them), this method aims to find the best predictor in terms of the lowest expected loss. The weights are calculated as posterior probability that each model is the one with the lowest loss. Bayesian model combination (BMC)[48] seeks the combination of models that is closest to the generating distribution by heavily weighting the most probable combination of models, instead of doing so for the most probable one. BMC samples from the space of possible ensembles by randomly drawing weights from a Dirichlet distribution with uniform parameters. The Bayesian Additive Regression Trees (BART)[50] method is one of the homogeneous ensemble approaches. It models the ensemble as a (nonweighted) sum of regression trees whose parameters, and ensemble error standard deviation, are defined thought their posterior distributions given data and sampled using MCMC. Yao et al.[49] suggest a predictive model in terms of a weighted combination of predictive distributions for each probabilistic model in the ensemble. This approach can be seen as a generalization of stacking for point estimation to predictive distributions.

All of these models, except of BMC and our model, have weights being point estimates, obtained usually by minimizing some error function. In contrast, we define them as random variables, and in contrast to BMC, our weights are defined through full joint posterior distribution given data. BMC is the closest in design to our approach, but it was formulated only in the context of classifiers. Only BART does include a random error term in the ensemble, apart from our model. Unlike BMA, BMC or models of Yao et al.[49], Chipman et al.[50], our approach does not require that the predictors are themselves probabilistic, and therefore can readily leverage various scikit-learn models. The main differences are summarized in Table S1.

Although our model has some features that have been previously considered in the literature, the approach presented here is, to the best of our knowledge, novel in the fact that the metalearner is modeled as a Bayesian linear regression model, whose parameters are inferred from data combined with a prior that satisfies the constraints on the 'voting' nature

Table S1: Feature differences between Bayesian based ensemble modeling approaches.

Method	Weighted average	Probabilistic base models	Probabilistic weights	Regression	Classification	Ensemble error term
BMA[47]	✔	✔	✘	✔	✔	✘
BMC[48]	✔	✔	✔✘	✘	✔	✘
BART[50]	✘	✔	✘	✔	✘	✔
Stacking predictive distributions[49]	✔	✔	✘	✔	✔	✘
Agnostic Bayes[89]	✔	✔✘	✘	✔	✔	✘
ART (this work)	✔	✘	✔	✔	✘	✔

of ensemble learners.

Input space set $\mathcal{B}$

The bounds for the input space $\mathcal{B}$ for $G(\mathbf{x})$ (Eq. 3) can be provided by the user (see details in the Implementation section, Table S3). Otherwise, default values are computed from the input data defining the feasible space as:

$$\begin{aligned} \mathcal{B} &= \{\tilde{\mathbf{x}} \in \mathbb{R}^D |\ \ L_d - \Delta_d \leq \tilde{x}^d \leq U_d + \Delta_d\ \ ,\ d = 1, \ldots, D\} \\ \Delta_d &= (U_d - L_d)\epsilon\ \ ;\ \ U_d = \max_{1 \leq n \leq N}(x_n^d)\ \ ;\ \ L_d = \min_{1 \leq n \leq N}(x_n^d) \\ &\qquad (\mathbf{x}_n, y_n) \in \mathcal{D},\ n = 1, \ldots, N \end{aligned} \tag{7}$$

The restriction of input variables to the set $\mathcal{B}$ reflects our assumption that the predictive models performs accurately enough only on an interval that is enlarged by a factor ϵ around the minimum and maximum values in the data set ($\epsilon = 0.05$ in all calculations).

Success probability calculation

Our probabilistic model enables us to estimate the probability of success for the provided recommendations. Of practical interest are the probability that a single recommendation is successful and the probability that at least one recommendation of several provided is successful.

Success is defined differently for each of the three cases considered in Eq. (3) : maximization, minimization and specification. For maximization, success involves obtaining a response y higher than the success value y^* defined by the user (e.g. the best production so far improved by a factor of 20%). For minimization, success involves obtaining a response lower than the success value y^*. For the specification case, it involves obtaining a response that is as close as possible to the success value y^*.

Formally, we define success for response y through the set $\mathcal{S} = \{y|y \sim p_{\mathcal{S}}(y)\}$, where the probability distribution for success is

$$p_{\mathcal{S}}(y) = \begin{cases} \mathcal{U}(y^*, U) & \text{(maximization case)} \\ \mathcal{U}(L, y^*) & \text{(minimization case)} \\ \mathcal{N}(y^*, \sigma^2_{y^*}) & \text{(specification case),} \end{cases} \tag{8}$$

where $\mathcal{U}$ is the uniform distribution ($\mathcal{U}(a, b) = 1/(b-a)$ if $a < y < b$; 0 otherwise), L and U are its corresponding lower and upper bounds, and $\sigma^2_{y^*}$ is the variance of the normal distribution $\mathcal{N}$ around the target value y^* for the specification case.

The probability that a recommendation succeeds is given by integrating the probability that input $\mathbf{x}^r$ gives a response y (i.e. full predictive posterior distribution from Eq. 2), times the probability that response y is a success

$$p(\mathcal{S}|\mathbf{x}^r) = \int p_{\mathcal{S}}(y) p(y|\mathbf{x}^r, \mathcal{D}) \mathrm{d}y.$$

This success probability is approximated using draws from the posterior predictive distribution as

$$p(\mathcal{S}|\mathbf{x}^r) \approx \begin{cases} \frac{1}{N_s} \sum_{i=1}^{N_s} \mathbb{I}_{\mathcal{S}}(y_i) & \text{(maximization/minimization case)} \\ \frac{1}{N_s} \sum_{i=1}^{N_s} \mathcal{N}(y_i; y^*, \sigma^2_{y^*}) & \text{(specification case)} \end{cases} \tag{9}$$

where $y_i \sim p(y|\mathbf{x}^r, \mathcal{D}), i = 1, \ldots, N_s$, and $\mathbb{I}_{\mathcal{A}}(y) = 1$ if $y \in \mathcal{A}$, 0 if $y \notin \mathcal{A}$.

In case of multiple recommendations $\{\mathbf{x}^r\} \equiv \{\mathbf{x}^r\}_{r=1}^{N_r}$, we provide the probability of success for at least one of the recommendations only for maximization and minimization types of objectives. This probability is calculated as one minus the probability $p(\mathcal{F}|\{\mathbf{x}^r\})$ that all recommendations fail, where

$$p(\mathcal{F}|\{\mathbf{x}^r\}) \approx \frac{1}{N_s}\sum_{i=1}^{N_s}\mathbb{I}_{\mathcal{F}}(\{y_i^r\}),\ \{y_i^r\} \sim p(y|\{\mathbf{x}^r\},\mathcal{D}), i = 1,\ldots,N_s, r = 1,\ldots,N_r,$$

and the *failure* set $\mathcal{F} = \{\{y^r\}|y^r \notin \mathcal{S}, \forall r = 1,\ldots,N_r\}$ consists of outcomes that are not successes for all of the recommendations. Since the chosen recommendations are not necessarily independent, we sample $\{y_i^r\}$ jointly for all $\{\mathbf{x}^r\}$, i.e. i-th sample has the same model parameters ($w_i, \sigma_i, \varepsilon_{ij} \sim \mathcal{N}(0,\sigma_i^2)$ from Eq. 1) for all recommendations.

Multiple response variables

For multiple response variable problems (e.g. trying to hit a predetermined value of metabolite a and metabolite b simultaneously, as in the case of the hopless beer), we assume that the response variables are conditionally independent given input vector $\mathbf{x}$, and build a separate predictive model $p_j(y_j|\mathbf{x},\mathcal{D})$ for each variable $y_j, j = 1,\ldots,J$. We then define the objective function for the optimization phase as

$$G(\mathbf{x}) = (1-\alpha)\sum_{j=1}^{J}\mathbb{E}(y_j) + \alpha\sum_{j=1}^{J}\mathrm{Var}(y_j)^{1/2}$$

in case of maximization, and analogously adding the summation of expectation and variance terms in the corresponding functions for minimization and specification objectives (Eq. 3). The probability of success for multiple variables is then defined as

$$p(\mathcal{S}_1,\ldots,\mathcal{S}_J|\mathbf{x}) = \prod_{j=1}^{J}p(\mathcal{S}_j|\mathbf{x}^r)$$

Future work will address the limitation of the independence assumption and take into

account possible correlations among multiple response variables.

Implementation

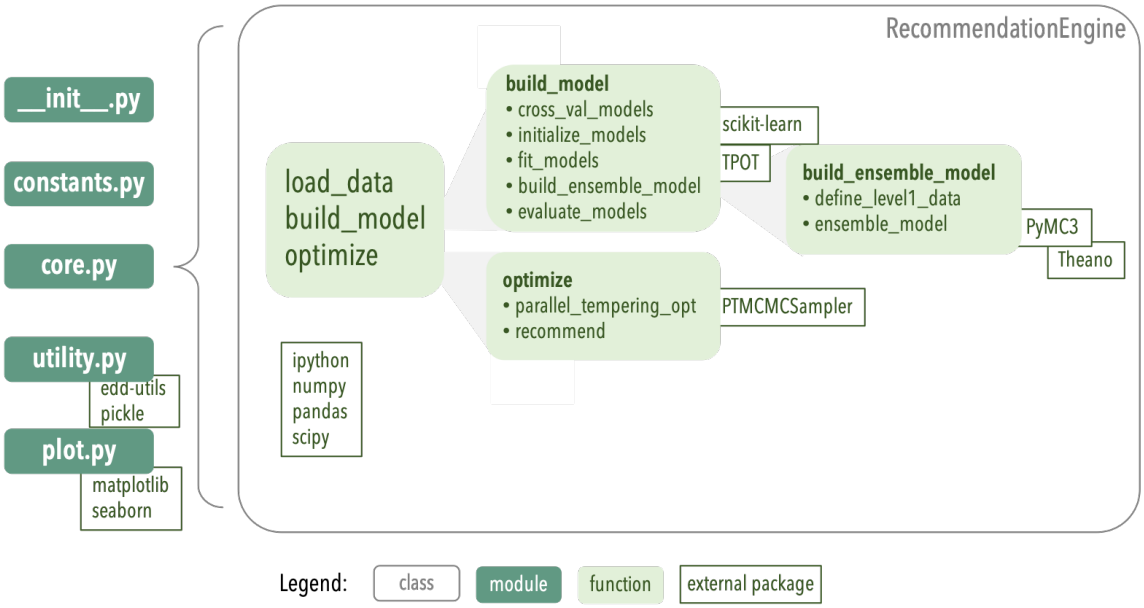


Figure S1: The main ART source code structure and dependencies.

Modules

`core.py` is the core module that defines the class `RecommendationEngine` with functions for loading data (into the format required for machine learning models), building predictive models and optimization (Fig. S1).

The module `constants.py` contains assignments to all constants appearing throughout all other modules. Those include default values for some of the optional user input parameters (Table S3), hyperparameters for `scikit-learn` models and simulation setups for `PyMC3` and `PTMCMCSampler` functions.

Module `utilities.py` is a suite of functions that facilitate ART's computations but can be used independently. It includes functions for loading studies (using EDD through

edd-utils or directly from files), metrics for evaluation of predictive models, identifying and filtering noisy data, etc.

Module plot.py contains a set of functions for visualization of different quantities obtained during an ART run, including functions of relevance to final users (e.g. true vs. predicted values) as well as those providing insights into intermediate steps (e.g. predictive models surfaces, space exploration from optimization, recommendations distributions).

All modules can be easily further extended by future contributors to ART.

Importing a study

Studies can be loaded directly from EDD by calling a function from the utility.py module that relies on edd-utils package:

```
dataframe = load_study(edd_study_slug=edd_study_slug,edd_server=edd_server)
```

The user should provide the study slug (last part of the study web address) and the url to the EDD server. Alternatively, a study can be loaded from an EDD-style .csv file, by providing a path to the file and calling the same function:

```
dataframe = load_study(data_file=data_file)
```

The .csv file should have the same format as an export file from EDD, i.e. it should contain at least Line Name column, Measurement Type column with names of input and response variables, and Value column with numerical values for all variables.
Either approach will return a pandas dataframe containing all information in the study, which can be pre-processed before running ART, if needed.

Running ART

ART can be run by instantiating an object from the RecommendationEngine class by:

```
art = RecommendationEngine(dataframe, **art_params)
```

The first argument is the dataframe created in the previous step (from an EDD study or data file import). If there is no data preprocessing, the dataframe is ready to be passed

as an argument. Otherwise, the user should make sure that the dataframe contains at least the required columns: `Line Name`, `Measurement Type` and `Value`. Furthermore, line names should always contain a hyphen ("-") denoting replicates (see Table S2), and this character should be exclusively used for this purpose (this point is critical for creating partitions for cross-validation).

Table S2: Valid and non valid examples of entries of the `Line Name` column in the dataframe passed to start an ART run.

✔ Valid	✘ Non valid
`LineNameX-1`	`LineNameX1`
`LineNameX-2`	`LineNameX2`
`LineNameX-r1`	`Line-NameX1`
`LineNameX-r2`	`Line-NameX2`
`LineNameX-R1`	`Line-Name-X1`
`LineNameX-R2`	`Line-Name-X2`
...	...

The second argument is a dictionary of key-value pairs defining several required and optional keyword arguments (summarized in Table S3) for generation of an `art` object.

Building the model

The level-0 models are first initialized and then fitted through the `_initialize_models` and `_fit_models` functions respectively, which rely on the `scikit-learn` and `tpot` packages. To build the final predictive model, first the level-1 data is created by storing cross-validated predictions of level-0 models into a `theano` variable that is shared across the functions from the `PyMC3` package. Finally, the parameters of the ensemble model are sampled within the function `_ensemble_model`, which stores the inferred model and traces that are later used for predictive posterior probability calculation, as well as first and second moments from the traces, used for estimation of the first two moments of the predictive posterior distribution using Eq. (5)–(6).

By default, ART builds the models using all available data and evaluates the final, ensemble model, as well as all level-0 models, on the same data. Optionally, if specified by the user through the input flag `cross_val`, ART will evaluate the models on 10-fold cross-validated

Table S3: ART input parameters. Required parameters are marked with an asterisk.

Name	Meaning
`input_var`	List of input variables*
`bounds_file`	Path to the file with upper and lower bounds for each input variable (default `None`)
`response_var`	List of response variables*
`build_model`	Flag for building a predictive model (default `True`)
`cross_val`	Flag for performing cross-validation (default `False`)
`ensemble_model`	Type of the ensemble model (default `'BW'`)
`num_models`	Number of level-0 models (default 8)
`recommend`	Flag for performing optimization and providing recommendations (default `True`)
`objective`	Type of the objective (default `'maximize'`)
`threshold`	Relative threshold for defining success (default 0)
`target_value`	Target value for the specification objective (default `None`)
`num_recommendations`	Number of recommendations for the next cycle (default 16)
`rel_eng_accuracy`	Relative engineering accuracy or required relative distance between recommendations (default 0.2)
`niter`	Number of iterations to use for $T = 1$ chain in parallel tempering (default 100000)
`alpha`	Parameter determining the level of exploration during the optimization (value between 0 and 1, default `None` corresponding to 0)
`output_directory`	Path of the output directory with results (default `../results/response_var_time_suffix`)
`verbose`	Amount of information displayed (default 0)
`seed`	Random seed for reproducible runs (default `None`)

predictions, through the function `_cross_val_models`. This computation lasts roughly 10 times longer. Evaluating models on new data, unseen by the models, can also be done by calling:

```
art.evaluate_models(X=X_new, y=y_new)
```

Optimization

ART performs optimization by first creating a set of draws from

```
draws = art.parallel_tempering_opt()
```

which relies on the `PTMCMCSampler` package. Here, an object from the class `TargetModel` is created. This class provides a template for and can be replaced by other types of objective functions (or target distributions) for parallel tempering type of optimization, as long as

it contains functions defining loglikelihood and logprior calculation (see Eq. 4). Also, the whole optimization procedure may well be replaced by an alternative routine. For example, if the dimension of the input space is relatively small, a grid search could be performed, or even evaluation of the objective at each point for discrete variables. Lastly, out of all draws collected by optimizing the specified objective, ART finds a set of recommendations by

```
art.recommend(draws)
```

which ensures that each recommendation is different from all others and all input data by a factor of γ (`rel_eng_accuracy`) in at least one of the components (see Algorithm 1).

Algorithm 1 Choosing recommendations from a set of samples from the target distribution $\pi(\mathbf{x})$

Input: N_r: number of recommendations
$\{\mathbf{x}_n\}_{n=1}^{N_s}$: samples from $\pi(\mathbf{x})$ (Eq. 4)
γ: required relative distance for recommendations (relative engineering accuracy)
$\mathcal{D}_{\mathbf{x}}$: input variable experimental data
Output: $rec = \{\mathbf{x}^r\}_{r=1}^{N_r}$: set of recommendations
$draws \leftarrow \{\mathbf{x}_n\}_{n=1}^{N_s}$ {remaining draws}
$rec = \emptyset$
while $r = 1, \ldots, N_r$ **do**
  if $draws = \emptyset$ **then**
    $\gamma = 0.8\gamma$ and repeat the procedure
  else
    $\mathbf{x}^r \leftarrow$ a sample from $draws$ with maximal $G(\mathbf{x})$ {$G(\mathbf{x})$ is already calculated}
    if there exists $d \in \{1, \ldots, D\}$ s.t. $|x_d^r - x_d| > \gamma$ for all $\mathbf{x} \in rec \cup \mathcal{D}_{\mathbf{x}}$ **then**
      $rec = \{rec, \mathbf{x}^r\}$
    end if
  end if
  $draws \leftarrow draws \setminus \{\mathbf{x}_n \in draws | G(\mathbf{x}_n) = G(\mathbf{x}^r)\}$
end while
return rec

Documentation and testing

The ART source code thoroughly conforms to the syntactic and stylistic specifications outlined in the PEP 8 style guide.[90] Documentation for class methods and utility functions is embedded in docstrings. A brief synopsis of functionality, an optional deeper dive into

behavior and use cases, as well as a full description of parameter and return value typing is included in the flexible reStructuredText markup syntax. The documented source code is used in conjunction with the `Sphinx` package to dynamically generate an API reference.

The code can be built as a local package for testing and other utility, the dependencies and procedure for which are handled by the `setup.py` file in accordance with the Setuptools standard for module installation.

A suite of unit and integration tests were written using the `pytest` library, and are included with the source code under the `tests` directory. The unit testing was designed to target and rigorously validate individual functions in their handling and output of types. Because of the expensive calls to libraries such as `scikit-learn` and `tpot` laden throughout ART's codebase for model training and validation, unit-tested functions were parameterized with the Boolean flag `testing` to replace such calls with dummy data that mimic the responses from library code. The resulting suite of unit tests therefore runs rapidly, quickly evaluating the integrity of type handling and output that ensure the smooth hand-off of values between functions in ART.

Integration tests are likewise implemented with `pytest`, but rather test a more complete spectrum of expected behavior without silencing calls to library code. Full model training, cross-validation, and evaluation is completed under this test suite, the output of which is validated against known data sets within some reasonable margin to account for stochastic processes.

The instructions to locally run the testing suite can be found in the documentation.

Supplementary figures and tables

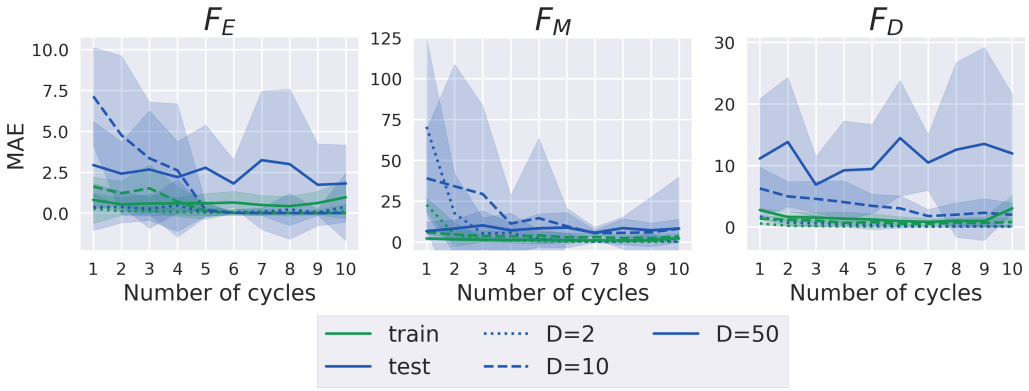


Figure S2: **Mean Absolute Error (MAE) for the synthetic data set in Fig. 4**. Synthetic data is obtained from functions of different levels of complexity (see Table 1), different phase space dimensions (2, 10 and 50), and different amounts of training data (DBTL cycles). The training set involves all the strains from previous DBTL cycles. The test set involves the recommendations from the current cycle. MAE are obtained by averaging the absolute difference between predicted and actual production levels for these strains. MAE decreases significantly as more data (DBTL cycles) are added, with the exception of the high dimension case. In each plot, lines and shaded areas represent the estimated mean values and 95% confidence intervals, respectively, over 10 repeated runs.

Table S4: Total number of strains (pathway designs) and training instances available for the dodecanol production study[74] (Figs. 8 , S5 and S6). Pathway 1, 2 and 3 refer to the top, medium and bottom pathways in Fig. 1B of Opgenorth et al.[74]. Training instances are amplified by the use of fermentation replicates. Failed constructs (3 in each cycle, initial designs were for 36 and 24 strains in cycle 1 and 2) indicate nontarget, possibly toxic, effects related to the chosen designs. Numbers in parentheses () indicate cases for which no product (dodecanol) was detected.

	Number of strains		Number of instances	
	Cycle 1	Cycle 2	Cycle 1	Cycle 2
Pathway 1	12	11 (2)	50	39 (6)
Pathway 2	9 (4)	10 (5)	31 (10)	30 (14)
Pathway 3	12	-	35	-
Total	33 (4)	21 (7)	116 (10)	69 (20)

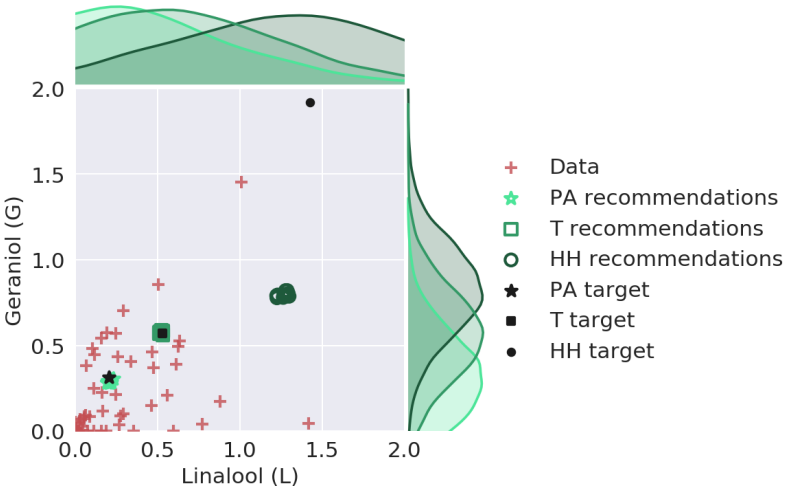


Figure S3: **Linalool and geraniol predictions for ART recommendations for each of the beers (Fig. 7), showing full probability distributions (not just averages)**. These probability distributions (in different tones of green for each of the three beers) show very broad spreads, belying the illusion of accurate predictions and recommendations. These broad spreads indicate that the model has not converged yet and that many production levels are compatible with a given protein profile.

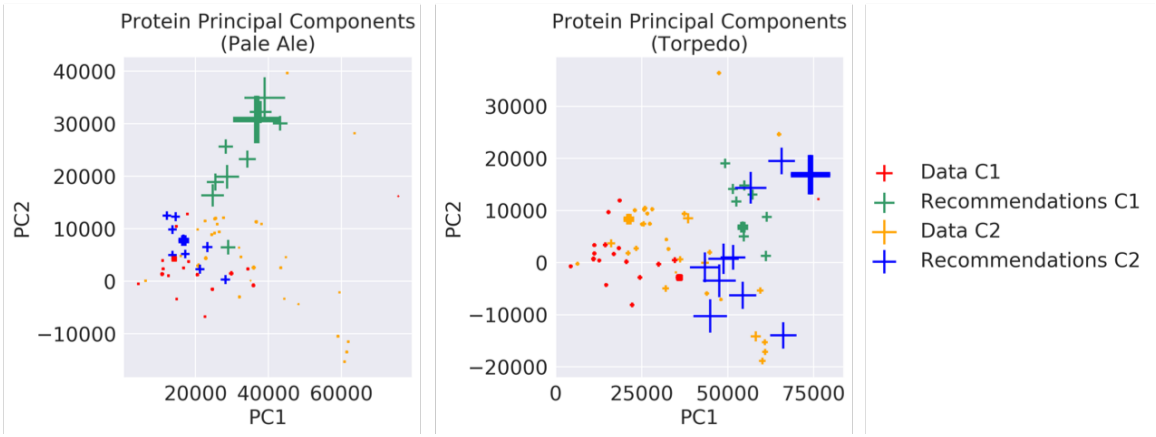


Figure S4: **Principal Component Analysis (PCA) of proteomics data for the hopless beer project (Fig. 7)**, showing experimental results for cycle 1 and 2, as well as ART recommendations for both cycles. Cross size is inversely proportional to proximity to L and G targets (larger crosses are closer to target). The broad probability distributions spreads (Fig. S3) suggest that recommendations will change significantly with new data. Indeed the protein profile recommendations for the Pale Ale changed markedly from DBTL cycle 1 to 2, even though the average metabolite predictions did not (Fig. 7, right column). For the Torpedo case, the final protein profile recommendations overlapped with the experimental protein profiles from cycle 2, although they did not cluster around the closest profile (largest orange cross), concentrating on a better solution according to the model. In any case, despite the limited predictive power afforded by the cycle 1 data, ART produces recommendations that guide the metabolic engineering effectively. For both of these cases, ART recommends exploring parts of the phase space such that the final protein profiles that were deemed close enough to the targets (in orange, see also bottom right of Fig. 7) lie between the first cycle data (red) and these recommendations (green). In this way, finding the final target (expected around the orange cloud) becomes an interpolation problem, which is easier to solve than an extrapolation one.

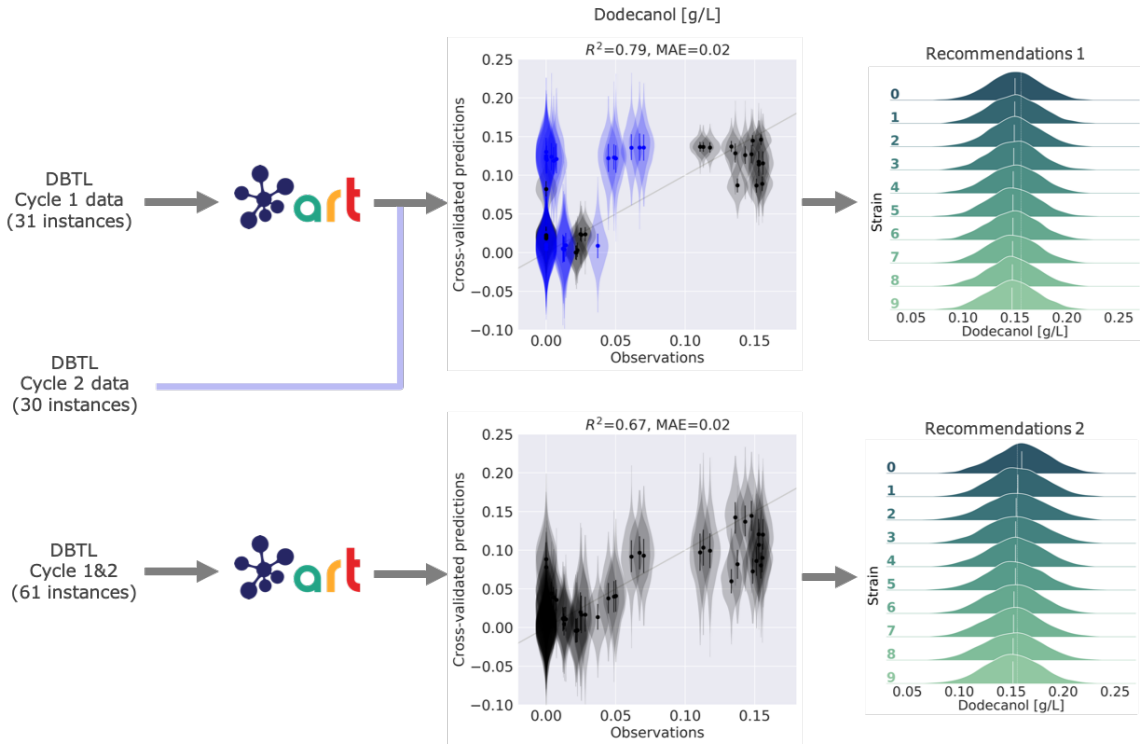


Figure S5: **ART's predictive power for the second pathway in the dodecanol production example is very limited.** Although cycle 1 data provide good cross-validated predictions, testing the model with 30 new instances from cycle 2 (in blue) shows limited predictive power and generalizability. As in the case of the first pathway (Fig. 8), combining data from cycles 1 and 2 improves predictions significantly.

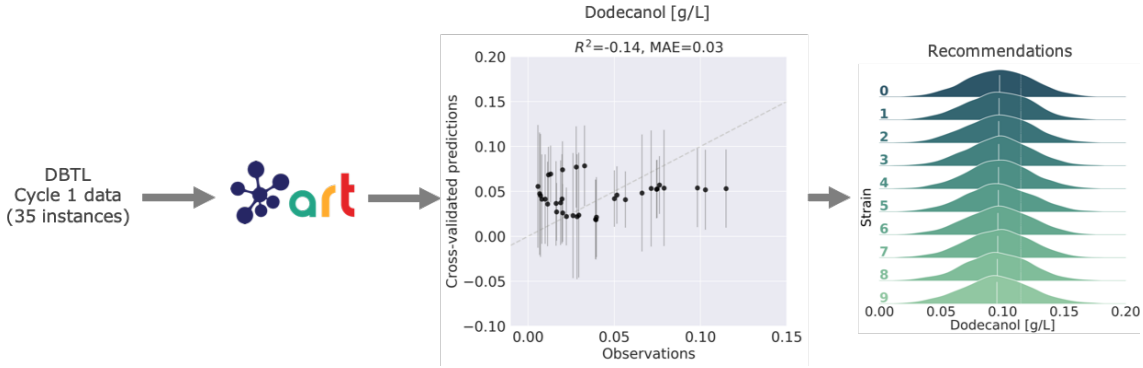


Figure S6: **ART's predictive power for the third pathway in the dodecanol production example is poor**. As in the case of the first pathway (Fig. 8), the predictive power using 35 instances is minimal. The low production for this pathway (Fig. 2 in Opgenorth et al.[74]) preempted a second cycle.

References

(1) Stephanopoulos, G. Metabolic fluxes and metabolic engineering. *Metabolic engineering* **1999**, *1*, 1–11.

(2) Beller, H. R.; Lee, T. S.; Katz, L. Natural products as biofuels and bio-based chemicals: fatty acids and isoprenoids. *Natural product reports* **2015**, *32*, 1508–1526.

(3) Chubukov, V.; Mukhopadhyay, A.; Petzold, C. J.; Keasling, J. D.; Martín, H. G. Synthetic and systems biology for microbial production of commodity chemicals. *npj Systems Biology and Applications* **2016**, *2*, 16009.

(4) Ajikumar, P. K.; Xiao, W.-H.; Tyo, K. E.; Wang, Y.; Simeon, F.; Leonard, E.; Mucha, O.; Phon, T. H.; Pfeifer, B.; Stephanopoulos, G. Isoprenoid pathway optimization for Taxol precursor overproduction in Escherichia coli. *Science* **2010**, *330*, 70–74.

(5) Cann, O. These are the top 10 emerging technologies of 2016. World Economic Forum website https://www.weforum.org/agenda/2016/06/top-10-emergingtechnologies-2016. 2016.

(6) National Research Council, *Industrialization of Biology: A Roadmap to Accelerate the Advanced Manufacturing of Chemicals*; National Academies Press, 2015.

(7) Yadav, V. G.; De Mey, M.; Lim, C. G.; Ajikumar, P. K.; Stephanopoulos, G. The future of metabolic engineering and synthetic biology: towards a systematic practice. *Metabolic engineering* **2012**, *14*, 233–241.

(8) Hodgman, C. E.; Jewett, M. C. Cell-free synthetic biology: thinking outside the cell. *Metabolic engineering* **2012**, *14*, 261–269.

(9) Kurian, J. V. A new polymer platform for the futureâĂŤSorona® from corn derived 1, 3-propanediol. *Journal of Polymers and the Environment* **2005**, *13*, 159–167.

(10) Cameron, D. E.; Bashor, C. J.; Collins, J. J. A brief history of synthetic biology. *Nature Reviews Microbiology* **2014**, *12*, 381.

(11) Kyrou, K.; Hammond, A. M.; Galizi, R.; Kranjc, N.; Burt, A.; Beaghton, A. K.; Nolan, T.; Crisanti, A. A CRISPR–Cas9 gene drive targeting doublesex causes complete population suppression in caged Anopheles gambiae mosquitoes. *Nature biotechnology* **2018**, *36*, 1062.

(12) Temme, K.; Tamsir, A.; Bloch, S.; Clark, R.; Emily, T.; Hammill, K.; Higgins, D.; Davis-Richardson, A. Methods and compositions for improving plant traits. 2019; US Patent App. 16/192,738.

(13) Chen, Y.; Guenther, J. M.; Gin, J. W.; Chan, L. J. G.; Costello, Z.; Ogorzalek, T. L.; Tran, H. M.; Blake-Hedges, J. M.; Keasling, J. D.; Adams, P. D.; Garcia Martin, H.; Hillson, N. J.; Petzold, C. J. Automated "Cells-To-Peptides" Sample Preparation Workflow for High-Throughput, Quantitative Proteomic Assays of Microbes. *Journal of proteome research* **2019**, *18*, 3752–3761.

(14) Fuhrer, T.; Zamboni, N. High-throughput discovery metabolomics. *Current opinion in biotechnology* **2015**, *31*, 73–78.

(15) Stephens, Z. D.; Lee, S. Y.; Faghri, F.; Campbell, R. H.; Zhai, C.; Efron, M. J.; Iyer, R.; Schatz, M. C.; Sinha, S.; Robinson, G. E. Big data: astronomical or genomical? *PLoS biology* **2015**, *13*, e1002195.

(16) Ma, S.; Tang, N.; Tian, J. DNA synthesis, assembly and applications in synthetic biology. *Current opinion in chemical biology* **2012**, *16*, 260–267.

(17) Doudna, J. A.; Charpentier, E. The new frontier of genome engineering with CRISPR-Cas9. *Science* **2014**, *346*, 1258096.

(18) Cumbers, J. Synthetic Biology Has Raised $12.4 Billion. Here Are Five Sectors It Will Soon Disrupt. 2019; https://www.forbes.com/sites/johncumbers/2019/09/04/synthetic-biology-has-raised-124-billion-here-are-five-sectors-it-will-soon-disrupt#40b2b2cb3a14.

(19) Petzold, C. J.; Chan, L. J. G.; Nhan, M.; Adams, P. D. Analytics for metabolic engineering. *Frontiers in bioengineering and biotechnology* **2015**, *3*, 135.

(20) Nielsen, J.; Keasling, J. D. Engineering cellular metabolism. *Cell* **2016**, *164*, 1185–1197.

(21) Gardner, T. S. Synthetic biology: from hype to impact. *Trends in biotechnology* **2013**, *31*, 123–125.

(22) Prinz, F.; Schlange, T.; Asadullah, K. Believe it or not: how much can we rely on published data on potential drug targets? *Nature reviews Drug discovery* **2011**, *10*, 712.

(23) Baker, M. 1,500 scientists lift the lid on reproducibility. *Nature News* **2016**, *533*, 452.

(24) Begley, C. G.; Ellis, L. M. Drug development: Raise standards for preclinical cancer research. *Nature* **2012**, *483*, 531.

(25) Carbonell, P.; Radivojević, T.; Martin, H. G. Opportunities at the Intersection of Synthetic Biology, Machine Learning, and Automation. *ACS Synth. Biol.* **2019**, *8*, 1474–1477.

(26) Thrun, S. Toward robotic cars. *Communications of the ACM* **2010**, *53*, 99–106.

(27) Wu, Y. et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* **2016**,

(28) Kosinski, M.; Stillwell, D.; Graepel, T. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences* **2013**, *110*, 5802–5805.

(29) Costello, Z.; Martin, H. G. A machine learning approach to predict metabolic pathway dynamics from time-series multiomics data. *NPJ systems biology and applications* **2018**, *4*, 19.

(30) Jervis, A. J. et al. Machine learning of designed translational control allows predictive pathway optimization in Escherichia coli. *ACS synthetic biology* **2018**, *8*, 127–136.

(31) Esteva, A.; Kuprel, B.; Novoa, R. A.; Ko, J.; Swetter, S. M.; Blau, H. M.; Thrun, S. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* **2017**, *542*, 115.

(32) Paeng, K.; Hwang, S.; Park, S.; Kim, M. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*; Springer, 2017; pp 231–239.

(33) Alipanahi, B.; Delong, A.; Weirauch, M. T.; Frey, B. J. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature biotechnology* **2015**, *33*, 831.

(34) Shaked, I.; Oberhardt, M. A.; Atias, N.; Sharan, R.; Ruppin, E. Metabolic network prediction of drug side effects. *Cell systems* **2016**, *2*, 209–213.

(35) Yang, J. H.; Wright, S. N.; Hamblin, M.; McCloskey, D.; Alcantar, M. A.; Schrübbers, L.; Lopatkin, A. J.; Satish, S.; Nili, A.; Palsson, B. O.; Walker, G. C.; Collins, J. J. A White-Box Machine Learning Approach for Revealing Antibiotic Mechanisms of Action. *Cell* **2019**, *177*, 1649–1661.

(36) Metz, C. AI Researchers Are Making More Than $1 Million, Even at a Nonprofit. *The New York Times* **2018**,

(37) Pedregosa, F. et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research* **2011**, *12*, 2825–2830.

(38) Gelman, A.; Carlin, J. B.; Stern, H. S.; Rubin, D. B. *Bayesian Data Analysis*, 2nd ed.; Chapman & Hall / CRC, 2003.

(39) Batth, T. S.; Singh, P.; Ramakrishnan, V. R.; Sousa, M. M.; Chan, L. J. G.; Tran, H. M.; Luning, E. G.; Pan, E. H.; Vuu, K. M.; Keasling, J. D.; Adams, P. D.; Petzold, C. J. A targeted proteomics toolkit for high-throughput absolute quantification of Escherichia coli proteins. *Metabolic engineering* **2014**, *26*, 48–56.

(40) Heinemann, J.; Deng, K.; Shih, S. C.; Gao, J.; Adams, P. D.; Singh, A. K.; Northen, T. R. On-chip integration of droplet microfluidics and nanostructure-initiator mass spectrometry for enzyme screening. *Lab on a Chip* **2017**, *17*, 323–331.

(41) HamediRad, M.; Chao, R.; Weisberg, S.; Lian, J.; Sinha, S.; Zhao, H. Towards a fully automated algorithm driven platform for biosystems design. *Nature Communications* **2019**, *10*, 1–10.

(42) Häse, F.; Roch, L. M.; Aspuru-Guzik, A. Next-generation experimentation with self-driving laboratories. *Trends in Chemistry* **2019**,

(43) Morrell, W. C. et al. The Experiment Data Depot: A Web-Based Software Tool for Biological Experimental Data Storage, Sharing, and Visualization. *ACS Synth. Biol.* **2017**, *6*, 2248–2259.

(44) Wolpert, D. The Lack of A Priori Distinctions between Learning Algorithms. *Neural Computation* **1996**, *8*, 1341–1390.

(45) Ho, T. K. Random Decision Forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition* **1995**,

(46) van der Laan, M.; Polley, E.; Hubbard, A. Super Learner. *Statistical Applications in Genetics and Molecular Biology* **2007**, *6*.

(47) Hoeting, J. A.; Madigan, D.; Raftery, A. E.; Volinsky, C. T. Bayesian model averaging: a tutorial. *Statistical Science* **1999**, *14*, 382–417.

(48) Monteith, K.; Carroll, J. L.; Seppi, K.; Martinez, T. Turning Bayesian model averaging into Bayesian model combination. The 2011 International Joint Conference on Neural Networks. 2011.

(49) Yao, Y.; Vehtari, A.; Simpson, D.; Gelman, A. Using Stacking to Average Bayesian Predictive Distributions (with Discussion). *Bayesian Analysis* **2018**, *13*, 917–1003.

(50) Chipman, H. A.; George, E. I.; McCulloch, R. E. Bayesian Ensemble Learning. Proceedings of the 19th International Conference on Neural Information Processing Systems. 2006; pp 265–272.

(51) Begoli, E.; Bhattacharya, T.; Kusnezov, D. The need for uncertainty quantification in machine-assisted medical decision making. *Nature Machine Intelligence* **2019**, *1*, 20.

(52) Olson, R. S.; Urbanowicz, R. J.; Andrews, P. C.; Lavender, N. A.; Kidd, L. C.; Moore, J. H. In *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30–April 1, 2016, Proceedings, Part I*; Squillero, G., Burelli, P., Eds.; Springer International Publishing, 2016; Chapter Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, pp 123–137.

(53) Breiman, L. Stacked regressions. *Machine Learning* **1996**, *24*, 49–64.

(54) LeDell, E. Scalable Ensemble Learning and Computationally Efficient Variance Estimation. Ph.D. thesis, University of California, Berkeley, 2015.

(55) Aldave, R. Systematic Ensemble Learning and Extensions for Regression. Ph.D. thesis, Université de Sherbrooke, 2015.

(56) Brooks, S., Gelman, A., Jones, G., Meng, X.-L., Eds. *Handbook of Markov Chain Monte Carlo*; CRC press, 2011.

(57) Mockus, J. *Bayesian Approach to Global Optimization: Theory and Applications*, 1st ed.; Springer Netherlands, 1989.

(58) Snoek, J.; Larochelle, H.; Adams, R. P. Practical Bayesian Optimization of Machine Learning Algorithms. NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems. 2012; pp 2951–2959.

(59) Earl, D. J.; Deem, M. W. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics* **2005**, *7*.

(60) McKay, M. D.; Beckman, R. J.; Conover, W. J. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics* **1979**, *21*, 239–245.

(61) Unthan, S.; Radek, A.; Wiechert, W.; Oldiges, M.; Noack, S. Bioprocess automation on a Mini Pilot Plant enables fast quantitative microbial phenotyping. *Microbial cell factories* **2015**, *14*, 32.

(62) Langholtz, M.; Stokes, B.; Eaton, L. 2016 Billion-ton report: Advancing domestic resources for a thriving bioeconomy, Volume 1: Economic availability of feedstock. *Oak Ridge National Laboratory, Oak Ridge, Tennessee, managed by UT-Battelle, LLC for the US Department of Energy* **2016**, *2016*, 1–411.

(63) Renouard-Vallet, G.; Saballus, M.; Schmithals, G.; Schirmer, J.; Kallo, J.; Friedrich, K. A. Improving the environmental impact of civil aircraft by fuel cell technology: concepts and technological progress. *Energy & Environmental Science* **2010**, *3*, 1458–1468.

(64) Keasling, J. D. Manufacturing molecules through metabolic engineering. *Science* **2010**, *330*, 1355–1358.

(65) Tracy, N. I.; Chen, D.; Crunkleton, D. W.; Price, G. L. Hydrogenated monoterpenes as diesel fuel additives. *Fuel* **2009**, *88*, 2238–2240.

(66) Ryder, J. A. Jet fuel compositions. 2009; US Patent 7,589,243.

(67) Duetz, W.; Bouwmeester, H.; Van Beilen, J.; Witholt, B. Biotransformation of limonene by bacteria, fungi, yeasts, and plants. *Applied microbiology and biotechnology* **2003**, *61*, 269–277.

(68) Alonso-Gutierrez, J.; Chan, R.; Batth, T. S.; Adams, P. D.; Keasling, J. D.; Petzold, C. J.; Lee, T. S. Metabolic engineering of Escherichia coli for limonene and perillyl alcohol production. *Metabolic engineering* **2013**, *19*, 33–41.

(69) Paddon, C. J. et al. High-level semi-synthetic production of the potent antimalarial artemisinin. *Nature* **2013**, *496*, 528.

(70) Meadows, A. L. et al. Rewriting yeast central carbon metabolism for industrial isoprenoid production. *Nature* **2016**, *537*, 694.

(71) Alonso-Gutierrez, J.; Kim, E.-M.; Batth, T. S.; Cho, N.; Hu, Q.; Chan, L. J. G.; Petzold, C. J.; Hillson, N. J.; D.Adams, P.; Keasling, J. D.; Martin, H. G.; SoonLee, T. Principal component analysis of proteomics (PCAP) as a tool to direct metabolic engineering. *Metabolic Engineering* **2015**, *28*, 123–133.

(72) Denby, C. M.; Li, R. A.; Vu, V. T.; Costello, Z.; Lin, W.; Chan, L. J. G.; Williams, J.; Donaldson, B.; Bamforth, C. W.; Christopher J. Petzold, H. V. S.; Martin, H. G.; Keasling, J. D. Industrial brewing yeast engineered for the production of primary flavor determinants in hopped beer. *Nature Communications* **2018**, *9*, 965.

(73) https://www.crunchbase.com/organization/berkeley-brewing-science#section-overview.

(74) Opgenorth, P. et al. Lessons from Two DesignâĂŞBuildâĂŞTestâĂŞLearn Cycles of Dodecanol Production in Escherichia coli Aided by Machine Learning. *ACS Synth. Biol.* **2019**, *8*, 1337–1351.

(75) Magnuson, K.; Jackowski, S.; Rock, C. O.; Cronan, J. E. Regulation of fatty acid biosynthesis in Escherichia coli. *Microbiology and Molecular Biology Reviews* **1993**, *57*, 522–542.

(76) Salis, H. M.; Mirsky, E. A.; Voigt, C. A. Automated design of synthetic ribosome binding sites to control protein expression. *Nature biotechnology* **2009**, *27*, 946.

(77) Espah Borujeni, A.; Channarasappa, A. S.; Salis, H. M. Translation rate is controlled by coupled trade-offs between site accessibility, selective RNA unfolding and sliding at upstream standby sites. *Nucleic Acids Research* **2013**, *42*, 2646–2659.

(78) Bonde, M. T.; Pedersen, M.; Klausen, M. S.; Jensen, S. I.; Wulff, T.; Harrison, S.; Nielsen, A. T.; Herrgård, M. J.; Sommer, M. O. Predictable tuning of protein expression in bacteria. *Nature methods* **2016**, *13*, 233.

(79) Ham, T.; Dmytriv, Z.; Plahar, H.; Chen, J.; Hillson, N.; Keasling, J. Design, implementation and practice of JBEI-ICE: an open source biological part registry platform and tools. *Nucleic Acids Research* **2012**, *40*.

(80) Granda, J. M.; Donina, L.; Dragone, V.; Long, D.-L.; Cronin, L. Controlling an organic synthesis robot with machine learning to search for new reactivity. *Nature* **2018**, *559*, 377.

(81) Le, K.; Tan, C.; Gupta, S.; Guhan, T.; Barkhordarian, H.; Lull, J.; Stevens, J.;

Munro, T. A novel mammalian cell line development platform utilizing nanofluidics and optoelectro positioning technology. *Biotechnology progress* **2018**, *34*, 1438–1446.

(82) Iwai, K.; Ando, D.; Kim, P. W.; Gach, P. C.; Raje, M.; Duncomb, T. A.; Heinemann, J. V.; Northen, T. R.; Martin, H. G.; Hillson, N. J.; Adams, P. D.; Singh, A. K. Automated flow-based/digital microfluidic platform integrated with onsite electroporation process for multiplex genetic engineering applications. 2018 IEEE Micro Electro Mechanical Systems (MEMS). 2018; pp 1229–1232.

(83) Gach, P. C.; Shih, S. C.; Sustarich, J.; Keasling, J. D.; Hillson, N. J.; Adams, P. D.; Singh, A. K. A droplet microfluidic platform for automating genetic engineering. *ACS synthetic biology* **2016**, *5*, 426–433.

(84) Hayden, E. C. The automated lab. *Nature News* **2014**, *516*, 131.

(85) Kendall, A.; Gal, Y. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? NIPS'17 Proceedings of the 31st Conference on Neural Information Processing Systems. 2017.

(86) Kiureghian, A. D.; Ditlevsen, O. Aleatory or epistemic? Does it matter? *Structural Safety* **2009**, *31*, 105–112.

(87) Breiman, L. Bagging Predictors. *Machine Learning* **1996**, *24*, 123–140.

(88) Freund, Y.; Schapire, R. E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences* **1997**, *55*, 119–139.

(89) Lacoste, A.; Marchand, M.; Laviolette, F.; Larochelle, H. Agnostic Bayesian Learning of Ensembles. Proceedings of the 31st International Conference on Machine Learning. 2014; pp 611–619.

(90) van Rossum, G.; Warsaw, B.; Coghlan, N. Style Guide for Python Code. 2001; www.python.org/dev/peps/pep-0008/.