# A MACHINE LEARNING APPROACH FOR MODELING AND INVERSE DESIGN OF POLYMER BLENDS

Industry Mentor

Dr. Nicolae C. Iovanac, DuPont de Nemours, Inc.

University of Delaware CHEG/CIS 867-015 Instructors:

Prof. Arthi Jayaraman and Prof. Sunita Chandrasekaran

May 12, 2022


Soham Jariwala
PhD
Chemical & Biomolecular Engineering
University of Delaware

Claudio César Claros Olivares
PhD
Electrical & Computer Engineering
University of Delaware

Zijie Wu
PhD
Chemical & Biomolecular Engineering
University of Delaware

## Abstract

Generating data-driven models for industrial polymer blends is of growing interest as it enables quantitative estimates of physical properties where the exact chemistry and structure are unknown. We outline a data-driven modeling approach to aid the discovery of new polymer blends by relating their ingredients and physical properties. This approach involves two components—a *forward model* that uses a machine learning method to predict physical property given a formulation as input, and an *inverse model* that generates candidate formulations for desired property targets using optimization. We compared various machine learning methods (similarity-based, tree-based, and shrinkage-based machine learning methods) by training and evaluating them on an experimental dataset of 78 distinct polymer blend formulations. Each formulation contains several of the 28 total ingredients. The physical properties of interest are water absorption, hardness, and thermal conductivity. Using a nested cross-validation scheme, we found that similarity-based machine learning methods, such as Gaussian process regression (GPR) and k-nearest neighbors, are suitable for generating robust forward models of polymer blends. The inverse model was developed using forward model and a particle swarm optimization (PSO) routine. We utilized the uncertainty predictions from GPR models to construct a loss function that gives robust optimal formulation candidates. With a modified loss function, we showed that one can generate candidate formulations that meet multiple design target properties.

## 1 Introduction

Polymers have hierarchical structures and complex interactions that span several orders of magnitude in length and time scales. Although a range of behaviors can be obtained by manipulating the monomer chemistry and structure of polymers, optimal functionality and performance may be achieved in a more economical manner by using mixtures of several polymers and small molecules, also known as polymer blends. The advantages of polymer blends, such as tunability to improve performance and processability, flexibility in manufacturing, and lower costs, make them highly desirable for industrial applications [1].

For a polymer with well defined composition and processing conditions, one may be able to apply physics-driven simulation techniques, such as Monte Carlo (MC), molecular dynamics (MD), and density functional theory (DFT) to obtain useful quantitative relationships for structure, composition, and physico-chemical properties [2]. However, in the case of polymer blends, where several ingredients (small molecules or polymers) are added to the base polymer, obtaining a quantitative relationship from simulations may become computationally intractable for large number of ingredients. Moreover, in cases where the chemistry and structure of the ingredients are not well characterized, it may not be possible to determine the force fields and interaction potentials required for physics-based simulations. As a result, formulation design of polymer blends relies heavily on heuristics and empirical relationships obtained from experimental measurements. This alternative data-driven approach is also called polymer informatics [3, 4], where machine-learning methods and data-driven modeling can accelerate the discovery and development of new formulations, allowing one to screen candidate polymer formulations for desired target properties.

In this work, we apply a polymer informatics approach to predict three target properties of a polymer blend: water absorption, hardness, and thermal conductivity, using the weight fractions of aliased ingredients as covariates.

## 2 Project Goal

The overarching goal of this project is to generate novel formulations with desired target properties. To achieve this goal, the project is divided into two stages. The first stage is to train a model that maps the ratios of the ingredients of the polymer formulation to the the aforementioned properties (forward-design) such that one can use this model to obtain the properties of any given formulation. The second stage is to obtain a formulation that meets the magnitude of a desired property target (inverse design) using the forward model to guide the search of the formulation through an optimization procedure.

## 3 Computational approach

The first step in our approach is to find mappings between our formulations and their physical properties: water absorption, hardness, and thermal conductivity, by training a model with a collection of formulas with known physical properties. The polymer formulation dataset comprises of 78 distinct formulations[1] containing weight fractions of ingredients such as plasticizers, toughenners, antioxidants, flame retardants and stabilizers in addition to the base polymer and fillers (see Figures 5, 6, and 7 in the Appendix for a graphical respresentation of the dataset). The dataset is aliased to protect the intellectual property of DuPont. Therefore, the identities and structural characteristics of these ingredients are not included in the model. A generic identifier is assigned to each ingredient based on its type (`Polymer_1, Filler_3, Clarifier_1`, etc.). The target properties associated with each formulation are the following:

- *Water absorption* is a quantitative measure of water absorbed by the polymer blend, usually reported as a percent water per dry weight. Water absoprtion depends on the hydrophilic chemistry and diffusion properties of the polymer blend. Water is present in the free volumes of the polymer network and attaches to the polymer chains via hydrogen bonds. In addition to the type of polymers and additives used, temperature and exposure time also have an effect on water absorption [5].
- *Hardness* is defined as resistance to local plastic deformation. Experimentally, hardness is determined

[1] In this work, instances, samples, formulation and formulas are used interchangeably.

by measuring the depth of indentation against an applied load. Values are reported as dimensionless number on scales that are specific to the type of hardness test conducted. Barcol, Rockwell (E,M,R), and Shore tests are some of the hardness tests suitable for polymers. Hardness is strongly influenced by testing temperature and the presence of additives [6].

- *Thermal conductivity* is the ability of the polymer blend to conduct heat reported in terms of heat flux per temperature gradient (units mW/m.K). Thermal conductivity depends on the nanoscale structure of the polymer chains and additives as well as the presence of crystalline and amorphous domains. Processing plays a key role in determining the thermal conductivity of the end product [7].

### 3.1 Forward design

The objective of the forward design is to obtain a ML model for each of the three target properties, i.e, water absoprtion, hardness, and thermal conductivity. However, before training any model, it is important to preprocess the dataset to assign similar significance to every ingredient regardless of its absolute value in composition, as we expect some ingredients can have significant impact on properties of interest despite their small absolute proportion. Particularly, we use a standard scaling for each column of ingredients (hereafter called a *feature*) such that $\bar{X} = \frac{X-\mu}{\sigma}$, where $X$ is a feature, $\mu$ is the feature mean, $\sigma$ is the feature standard deviation (sample), and $\bar{X}$ is the standardized feature.

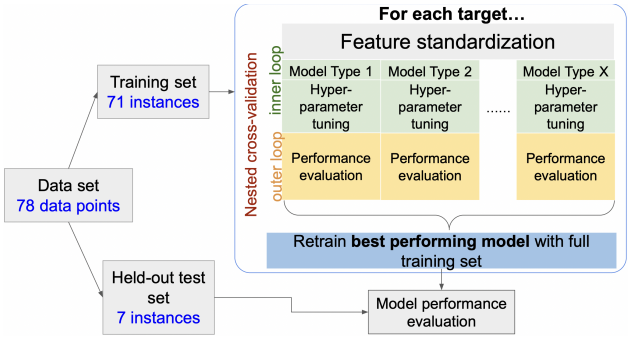



Figure 1: Forward design approach showing the scheme for model selection. Hyper-parameter tuning was performed using a nested cross-validation scheme for each model, followed by retraining the best performing model on the full 71-instance training set. The model performance was then tested against held-out dataset of 7 formulation instances.

Formally, each formulation $\mathbf{x}$ is an $M$-dimensional vector, i.e., $\mathbf{x} \in [0,1]^M$, where $M$ is the cardinality of the set of all ingredients (feature space). However, only $D < M$ components in each sample are different from zero and the sum of these $D$ elements is 1. This means that the remaining $M - D$ elements in this vector are zeros, which make the dataset sparse in the feature space.

Based on the sparsity of the feature space present in our dataset and the multi-task nature of this problem, we train multiple regression models that can exploit structures of the formulations to predict the target properties. To tackle sparse feature spaces, one possibility is to apply linear models that enforce sparsity in the coefficients that weigh each feature through regularization techniques such that these weights represent the relevance of each feature in the prediction [8]. The linear regression models that we use are Ridge, LASSO, and Elastic Net. Another possibility is to split the feature space using regression trees, and make predictions based on the aggregates of the instances that lie inside the regions generated by these trees [9]. The tree-based methods considered in this work are Gradient boosting and Random Forest regression. A third option is to leverage the similarity of the instances in the dataset and make predictions relying on a weighted combination of these similarities [10]. The similarity-based techniques applied in this work are k-Nearest Neighbors and Gaussian Process regression.

In order to select the best predictive model out of all the regressors considered in this work, we perform a nested cross-validation procedure (Figure 8), which tackles bias in performance evaluation[11]. In contrast to a traditional cross-validation execution, a nested cross-validation has an inner loop in which model hyper-parameters are tuned and an outer loop in which model performance is evaluated. By doing this, we avoid an overoptimistic evaluation that could mislead us in the model selection. The selection of the best model for each target also considered applying/not applying a variance-based feature selection mechanism,

applying/not applying standardization, and augmenting/not augmenting the original dataset with 2 features: mixing entropy $H(\mathbf{x}) = \sum_i x_i \log x_i$ and uncentered variance $V(\mathbf{x}) = \sum_i x_i^2$, where $\mathbf{x} = [x_1, \ldots, x_M]$.

Lastly, uncertainty associated to a prediction is crucially important for the reliability of ML models in materials science[12]. Particularly in our case, generating new formulations requires confidence intervals because this information exposes regions in the feature space where we do not have enough evidence for a confident prediction. In that sense, uncertainty can help us navigate the feature space to either exploit information about known formulations or explore possibly novel blends. Several approaches for estimating uncertainty such as bootstrapping, dropout for neural networks, and Bayesian methods[13–15] have been developed in order to provide prediction intervals for models that do not provide them natively. Nonetheless, Gaussian processes[16], out of all the models we use in training, provide uncertainty estimates for its predictions in a principled way. Strictly speaking, a Gaussian process $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, defined by a mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$, provides Gaussian predictive distributions $p(f_*|\mathbf{x}_*, X, \mathbf{y}) \sim \mathcal{N}(\bar{f}_*, \sigma(f_*))$ for $f_* = f(\mathbf{x}_*)$ at unseen instances $\mathbf{x}_*$, where $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ are the observed data instances, $\mathbf{y} = [y_1, \ldots, y_N]$ are the observed targets, and $N$ is the number of data instances.

### 3.2 Inverse Design

The objective of the inverse design is to propose candidate formulations that meet targets for each of the properties. In this work, we first narrow our focus to developing an inverse design scheme for a single property target, allowing us to reformulate it as a single-objective, constrained optimization problem, before briefly exploring multi-target optimization. The optimization algorithm we use is Particle Swarm Optimization (PSO) [17], a heuristic-based optimization method that is especially well suited for this purpose. It is relatively robust, has easy-to-tune hyper-parameters, requires little knowledge of the underlying structure of the optimized function, and does not depend on the gradient of the optimized function (see the appendix for more detailed description of the algorithm), the latter being especially important if the chosen model from the forward design step ends up being similarity-based and thus difficult to differentiate. We initialize the particles at the positions of the 71 training instances used for the forward design with random velocities to keep the particles close to these training samples, for which we have high confidence in the predictions from our models. We define the loss function, $\ell(\mathbf{x})$, as follows: $\ell(\mathbf{x}) = \left(\bar{f}(\mathbf{x}) - TARGET\right)^2 + \lambda \sigma^2(f(\mathbf{x}))$, where $\lambda$ is a scaling factor that controls the contribution from the variance term characterizing the certainty in our prediction, and $\bar{f}(\mathbf{x}), \sigma^2(f(x))$ are the evaluated mean value and variance by the forward model. By taking the variance term automatically provided by the Gaussian Process Regression forward model into consideration, this form of loss function balances minimizing deviation from the target and providing solutions in regimes of high prediction confidence. To maintain realistic solutions, the composition of each ingredient needs to be in the range $[0, 1)$, and the sum of compositions for all ingredients needs to be exactly one. To implement these constraints, we construct the feature vector with 27 out of the 28 ingredients, and search for each of the composition in the range of $[0, 1)$. The left-out 28-th ingredient, which is chosen to be *Polymer 1* due to its ubiquitous presence in bulk amount in most of the training instances, can then be automatically calculated as one minus sum of all other ingredients. When the sum of the 27-component feature vector exceeds 1, we do not use $\ell(\mathbf{x})$ to evaluate a loss function, and instead assign an arbitrarily large number ($10^5$) as the loss, which effectively prevents the optimizer from providing such feature vector as the candidate solution.

For multi-target optimzation, as an exploratory attempt we set the total loss function to the weighed sum of loss functions for individual targets: $\ell_{multi}(\mathbf{x}) = \sum_i w_i \ell_i(\mathbf{x})$, where $i$ is the target property.

## 4 Results & Discussion

The results of the nested cross-validation procedure incorporating a 10-fold outer loop and a 5-fold inner loop for model selection are displayed in Table 1 in the appendix. Running a nested cross-validation procedure is computationally expensive as hyper-parameters need to be trained for each model multiple times. Therefore, we execute the model selection procedure on DARWIN supercomputing cluster at UD. Specifi-

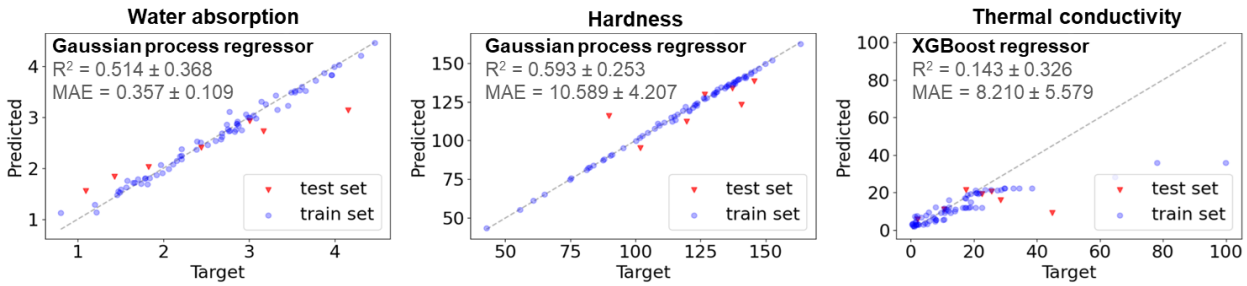



Figure 2: Predictions of the best performing models compared with the training data set (blue symbols) and test set values (red symbols), (a) water absorption, (b) hardness, (c) thermal conductivity.

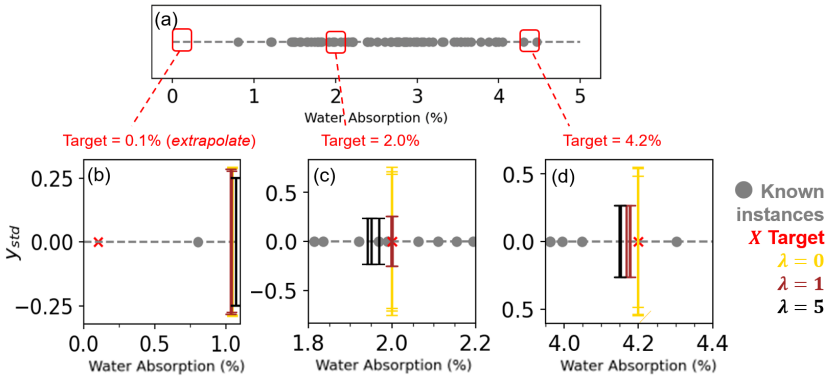



Figure 3: Results of single-target optimization for water absorption. (a) Distribution of all water absorption values of the training instances, (b-d) Predictions for single-target particle swarm optimization with water absorption target = (b) 0.1%, (c) 2.0%, (d) 4.2%. For each (target,$\lambda$) three repeated runs are performed and plotted with three distinct error bars (overlapping when not distinguishableable on the plot). In (b-d), target value is marked by red cross, predicted water absorption values of "best" found formulation are represented by the x-axis values of the colored error bars, and the standard deviation of the predicted distribution of water absorption is represented by error bar along the y-axis.

cally, we use 1 standard compute node and 10 CPUs to accelerate this process. The best performing models selected from the cross-validation procedure are Gaussian process regression (GPR) for water absorption, k-nearest neighbors for hardness, and XGBoost for thermal conductivity. We decide to also use GPR as the forward model for hardness as the performance metrics ($R^2$ and mean absolute error (MAE)) are close while GPR has the additional benefit of providing the uncertainty. The comparison of predictions and targets for selected models is shown in Figure 2

The model predictions are in reasonable agreement with the data for water absorption and hardness. However, for thermal conductivity, the selected model shows poor agreement. We suspect the underlying cause of the latter to be the varying processing conditions between instances, as thermal properties of polymers are known to be heavily influenced by the processing conditions. The distribution of amorphous and crystalline sub-phases can result in dramatically different thermal conductivity values for similar compositions [7]. Water absorption and hardness predictions are reasonably accurate considering the lack of chemistry, size, and structure of the ingredients in the dataset. Including additional information about the physical nature of the ingredients, such as monomer chemistry, sizes, molecular weights, may result in improved predictions, as reported in the literature for similar problems [3, 4, 18].

The predictions from single-target PSO for water absorption are shown in Figure 3. We highlight the effect of $\lambda$ in $\ell(\mathbf{x})$ in regularizing the predicted formulations within regimes of high confidence. When the

target value is within the range of those in training instances (Figure 3c,d), at $\lambda = 0$, no regularization is applied and a blend formulation with predicted water absorption value exactly equal to the target is obtained, albeit with significant uncertainty in prediction (shown by error bars and characterized by the standard deviation in predicted distribution automatically output by the GPR model). As $\lambda$ increases, the predictions in general deviate slightly from the target in exchange for a lower uncertainty in prediction. Since our forward models are similarity-based models built in a multi-dimensional parameter space with sparse training data, we expect the model to be reliable only in a small subspace close to the training instances, and therefore we believe that a predicted formulation that balances accuracy vs. certainty and resembles the known formulation compositions is more promising for experimental testing than a predicted formulation that has "exact" target values but unreliable uncertainty. In Figure 3b, we test an extrapolative case with target water absorption (0.1%) much lower than the range of values seen in training instances. We see that the optimizer is not able to find a formulation with water absorption at this target regardless of $\lambda$, and the predicted values generally stop at the lowest water absorption seen in training instances ( 1.0%). This result shows the expected inability of similarity-based models (i.e., GPR models) to extrapolate beyond seen training instances. In this circumstance of low amount of training data, we believe that it is desirable that our algorithms clearly signal and automatically avoid the region of low confidence during the inverse search. In Figure 9 in Appendix, we show search results with hardness as the target, leading to similar conclusions as we have discussed here for water absorption. In Figure 10 in Appendix, we show the effect of $\lambda$ on the consistency between predicted formulations of repeated runs for the same target value and $\lambda$. We see that in the interpolation regime, higher $\lambda$ also leads to more consistent predicted formulations.

In Figure 4, we show the results of an attempt of the dual-target optimization searching for a formulation that gives a certain water absorption *and* a certain hardness value. In this proof-of-concept attempt, we set $w_i = 1/TARGET_i$ in $\ell_{multi}(\mathbf{x})$ to give a qualitatively similar weight to the two target properties. In practice, one can tune the $w_i$ to distribute appropriate weight to each target depending on their actual significance. Similar to what we see in the single-target optimization, $\lambda$ plays the role of regularizing the found solution towards regimes of higher confidence, at the cost of lower absolute accuracy, shown by the shorter error bars characterizing lower uncertainty.

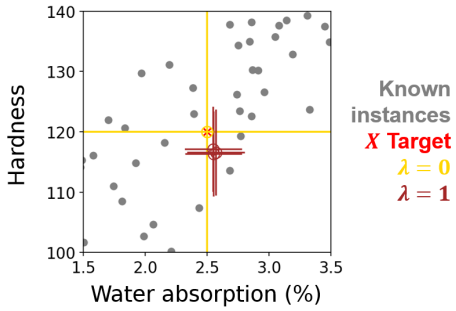



Figure 4: Results of multi-target optimization for target of water absorption=2.5% and Hardness=120. Three repeated runs are performed and plotted with three distinct error bars (overlapping when not distinguishable on the plot). Target value is marked by red cross, all training instances are marked by grey dots, and the standard deviation of the predicted distribution of respective targets are represented by error bar along the x-axis (water absorption) and y-axis (hardness).

## 5 Conclusions

We present an approach for modeling and inverse design of polymer blends to meet targets for three physical properties, namely, water absorption, hardness, and thermal conductivity. The data set used contains the proportions of aliased ingredients present in a given polymer blend. We found that forward predictions for water absorption and hardness using instance-based methods were in good agreement with the test data set. Thermal conductivity, which depends on polymer processing as well as composition showed poor agreement. We expect the predictions to improve if more chemical and structural information about the ingredients as well as processing conditions, such as temperature and stresses, are included in the dataset. Finally, we developed a particle swarm optimization scheme based on the forward models to predict candidate formulations that can aid in accelerated discovery of new blends. The incorporation of uncertainty

into the loss function of the optimizer helps us regularize the determined blends within regimes of high confidence.

## Data availability

The data set containing the 71 training instances as well as the code for the model selection procedure can be found in https://github.com/cesar-claros/DuPont.

## Acknowledgments

The authors gratefully acknowledge the University of Delaware and DuPont company for providing the opportunity to pursue this project. This work was performed as part of the course CHEG/CISC 867-015 (Spring 2022). The author express their thanks to the course instructors, Prof. Sunita Chandrasekaran and Prof. Arthi Jayaraman for their support and project mentor Dr. Nicolae C. Iovanac for guidance and instructive discussions.

This research was supported in part through the use of the NSF sponsored DARWIN computing system, award No: 1919839. DARWIN: A Resource for Computational and Data-intensive Research at the University of Delaware and in the Delaware Region, Rudolf Eigenmann, Benjamin E. Bagozzi, Arthi Jayaraman, William Totten, and Cathy H. Wu, University of Delaware, 2021, URL: https://udspace.udel.edu/handle/19716/29071.

## Author contributions

Authors contributions in alphabetical order:

Soham Jariwala (Conceptualization, methodology, forward design, inverse design, writing–original draft, editing)

Claudio César Claros Olivares (Conceptualization, methodology, forward design, inverse design, software–model selection, writing–original draft, editing)

Zijie Wu (Conceptualization, methodology, forward design, inverse design, software–optimization, PSO, writing–original draft, editing)

Dr. Nicolae C. Iovanac, Industry Mentor (Problem conceptualization, guidance and discussions on models and methodologies)

Dr. Arthi Jayaraman, Course Instructor (Computational resources, Editing – Original draft report, Final report)

Dr. Sunita Chandrasekaran (Computational resources, Editing – Final report)

## References

[1] E. Manias and L. A. Utracki, "Thermodynamics of polymer blends," in *Polymer Blends Handbook*, L. A. Utracki and C. A. Wilkie, Eds. Dordrecht: Springer Netherlands, 2014, pp. 171–289, ISBN: 978-94-007-6064-6. DOI: 10.1007/978-94-007-6064-6{_}4.

[2] T. E. Gartner and A. Jayaraman, "Modeling and simulations of polymers: A roadmap," *Macromolecules*, vol. 52, no. 3, pp. 755–786, Feb. 2019. DOI: 10.1021/acs.macromol.8b01836.

[3] D. J. Audus and J. J. de Pablo, "Polymer informatics: Opportunities and challenges," *ACS Macro Letters*, vol. 6, no. 10, pp. 1078–1082, Oct. 2017. DOI: 10.1021/acsmacrolett.7b00228.

[4] L. Chen *et al.*, "Polymer informatics: Current status and critical next steps," *Materials Science and Engineering: R: Reports*, vol. 144, p. 100 595, 2021. DOI: https://doi.org/10.1016/j.mser.2020.100595.

[5] G. Baschek, G. Hartwig, and F. Zahradnik, "Effect of water absorption in polymers at low and high temperatures," *Polymer*, vol. 40, no. 12, pp. 3433–3441, 1999. DOI: https://doi.org/10.1016/S0032-3861(98)00560-6.

[6] polymerdatabase.com. "Hardness of plastics and rubbers." (2022), [Online]. Available: https://polymerdatabase.com/polymer%20physics/ShoreRockwell.html.

[7] C. Huang, X. Qian, and R. Yang, "Thermal conductivity of polymers and polymer nanocomposites," *Materials Science and Engineering: R: Reports*, vol. 132, pp. 1–22, 2018. DOI: https://doi.org/10.1016/j.mser.2018.06.002.

[8] T. Hastie, R. Tibshirani, and M. Wainwright, "Statistical learning with sparsity," *Monographs on statistics and applied probability*, vol. 143, p. 143, 2015.

[9] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. Routledge, 2017.

[10] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The annals of statistics*, vol. 36, no. 3, pp. 1171–1220, 2008.

[11] G. C. Cawley and N. L. C. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *Journal of Machine Learning Research*, vol. 11, no. 70, pp. 2079–2107, 2010.

[12] F. Tavazza, B. DeCost, and K. Choudhary, "Uncertainty prediction for machine learning models of material properties," *ACS Omega*, vol. 6, no. 48, pp. 32 431–32 440, 2021. DOI: 10.1021/acsomega.1c03752. eprint: https://doi.org/10.1021/acsomega.1c03752.

[13] D. L. Shrestha and D. P. Solomatine, "Machine learning approaches for estimation of prediction interval for the model output," *Neural Networks*, vol. 19, no. 2, pp. 225–235, 2006, Earth Sciences and Environmental Applications of Computational Intelligence, ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2006.01.012.

[14] M. Abdar *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, 2021, ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2021.05.008.

[15] K. Vaysse and P. Lagacherie, "Using quantile regression forest to estimate uncertainty of digital soil mapping products," *Geoderma*, vol. 291, pp. 55–64, 2017, ISSN: 0016-7061. DOI: https://doi.org/10.1016/j.geoderma.2016.12.017.

[16] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.

[17] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[18] J. N. Kumar, Q. Li, K. Y. T. Tang, T. Buonassisi, A. L. Gonzalez-Oyarce, and J. Ye, "Machine learning enables polymer cloud-point engineering via inverse design," *npj Computational Materials*, vol. 5, no. 1, p. 73, 2019. DOI: 10.1038/s41524-019-0209-9.

[19] L. J. V. Miranda, "PySwarms, a research-toolkit for Particle Swarm Optimization in Python," *Journal of Open Source Software*, vol. 3, 21 2018. DOI: 10.21105/joss.00433.

## Appendix

### Particle swarm optimization (PSO) algorithm

The Particle swarm optimization starts with an initial "swarm" of particles with initial velocities and positions within the parameter space that are successively updated based on the quality of solutions previously visited until the optimality criterion is met. The inverse design scheme was developed in Python using PSO implementation in `PySwarms` package by Miranda et al. [19]. At successive iterations, each particle moves to its next position evaluated based on the best solution (the solution with smallest loss) visited locally by the particle and globally by the entire swarm in the previous iteration. Then at successive iterations, each particle moves to its next position evaluated based on the best solution (the solution with smallest loss) visited locally by the particle and globally by the entire swarm in the previous iteration:

$$\mathbf{v_{i,new}} = w\mathbf{v_{i,old}} + r_c c_c(\mathbf{x_{i,best}} - \mathbf{x_{i,old}}) + r_s c_s(\mathbf{g} - \mathbf{x_{i,old}}), \tag{1}$$

$$\mathbf{x_{i,new}} = \mathbf{x_{i,old}} + \mathbf{v_{i,old}}. \tag{2}$$

In these equations, $\mathbf{v_{i,new}}$ and $\mathbf{x_{i,new}}$ refer to the updated velocity and position of the partcle, $\mathbf{v_{i,old}}$ and $\mathbf{x_{i,old}}$ refer to the previous velocity and position of the particle; $w$, $c_c$ and $c_s$ are the three hyperparameters of PSO (set to $w = 0.2$,$c_c = 0.1$,$c_s = 0.06$ in this work), commonly named as "inertia", "cognitive" and "social" terms respectively, that together controls the rate of convergence and aggressiveness in exploration of unvisited parameter space; $r_c$ and $r_s$ are two uniform random variables sampled from $[0,1)$; $\mathbf{x_{i,best}}$ and $\mathbf{g}$ refer to the previous positions visited by this particle and the entire swarm, respectively, that minimizes the loss function.

## Additional figures and tables

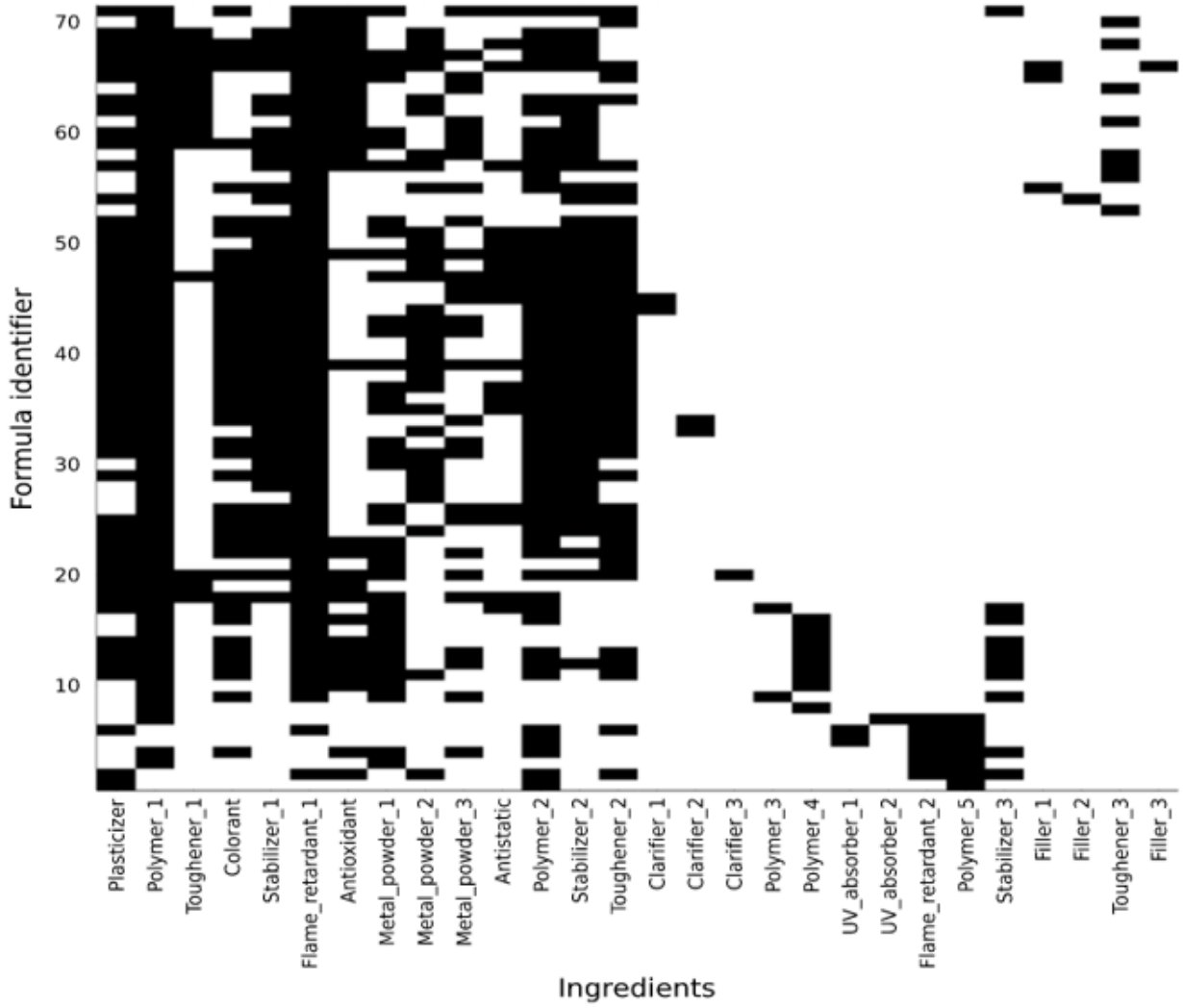



Figure 5: Visual representation of the dataset showing the ingredients of every formula. Non-zero values are indicated in black. The dataset is sparse, especially for the ingredients `Clarifier_1`, `Clarifier_2`, `Clarifier_3`, `Polymer_3`, `UV_absorber_1`, `UV_absorber_2`, `Filler_2`, and `Filler_3`, where fewer than 3 non-zero instances are present.

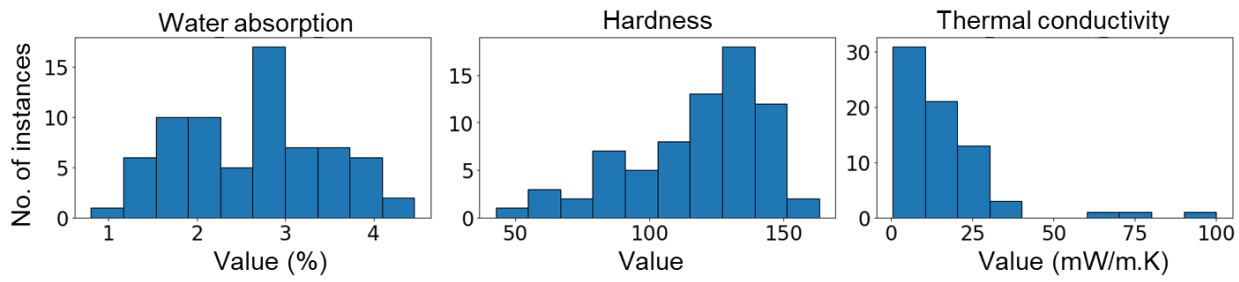



Figure 6: Histograms for each of the three target properties. Water absorption and hardness are distributed across the whole range of values. Thermal conductivity on the other hand is clustered primarily around the lower end of the values, with very few instances representing the higher end.

Table 1: Results of a nested cross-validation procedure for model selection. The models that perform the best in at least two out of the three metrics are highlighted in blue for each of the target properties. Results are shown as (mean,std).

| **Target** | **Model** | **R2** | **MAE** | **RMSE** |
|---|---|---|---|---|
| Water Absorption % | K-Neighbors Regressor | (0.404, 0.404) | (0.412, 0.074) | (0.282, 0.110) |
| | **Gaussian Process Regressor** | **(0.514, 0.368)** | **(0.357, 0.109)** | **(0.238, 0.170)** |
| | XGB Regressor | (0.443, 0.487) | (0.396, 0.097) | (0.267, 0.165) |
| | Random Forest Regressor | (0.327, 0.908) | (0.396, 0.100) | (0.276, 0.172) |
| | Ridge | (0.238, 0.521) | (0.475, 0.108) | (0.381, 0.202) |
| | Lasso | (0.176, 0.679) | (0.473, 0.107) | (0.411, 0.318) |
| | ElasticNet | (0.362, 0.538) | (0.417, 0.102) | (0.298, 0.164) |
| Hardness | **K-Neighbors Regressor** | **(0.593, 0.253)** | **(10.589, 4.207)** | (237.863, 244.553) |
| | Gaussian Process Regressor | (0.470, 0.4727) | (11.164, 4.084) | (252.464, 237.085) |
| | XGB Regressor | (0.473, 0.412) | (12.058, 5.766) | (294.037, 261.661) |
| | Random Forest Regressor | (0.546, 0.282) | (11.326, 4.259) | **(234.805, 178.271)** |
| | Ridge | (0.194, 0.433) | (14.511, 4.224) | (414.912, 300.542) |
| | Lasso | (0.380, 0.287) | (13.195, 3.575) | (309.369, 186.328) |
| | ElasticNet | (0.163, 0.715) | (13.964, 4.167) | (389.851, 271.575) |
| Thermal Conductivity (mW/m.K) | K-Neighbors Regressor | (0.000, 0.727) | (9.499, 6.807) | (265.546, 458.483) |
| | Gaussian Process Regressor | (-0.658, 2.070) | (10.230, 6.806) | (304.167, 456.758) |
| | **XGB Regressor** | **(0.143, 0.326)** | **(8.210, 5.579)** | (225.724, 361.156) |
| | Random Forest Regressor | (-0.108, 0.717) | (8.943, 5.372) | **(221.194, 350.100)** |
| | Ridge | (-0.083, 0.593) | (9.225, 6.356) | (240.918, 426.989) |
| | Lasso | (-0.053, 0.629) | (9.124, 6.198) | (239.485, 433.459) |
| | ElasticNet | (0.050, 0.526) | (8.736, 6.618) | (243.681, 439.969) |

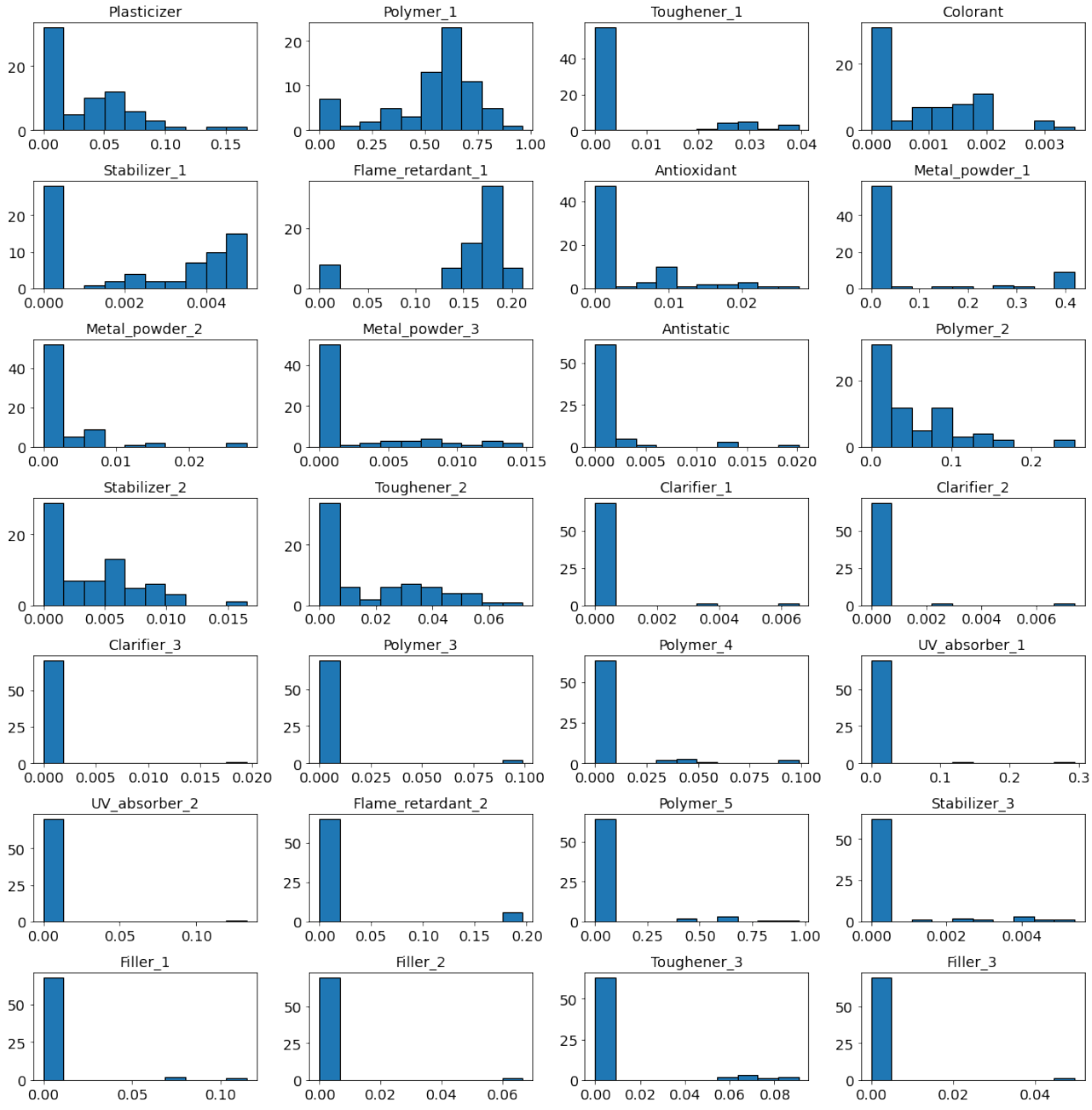



Figure 7: Histograms showing the distribution of all 28 ingredients in the dataset. For a majority of ingredients, there is a significant number of instances that have a value of 0.

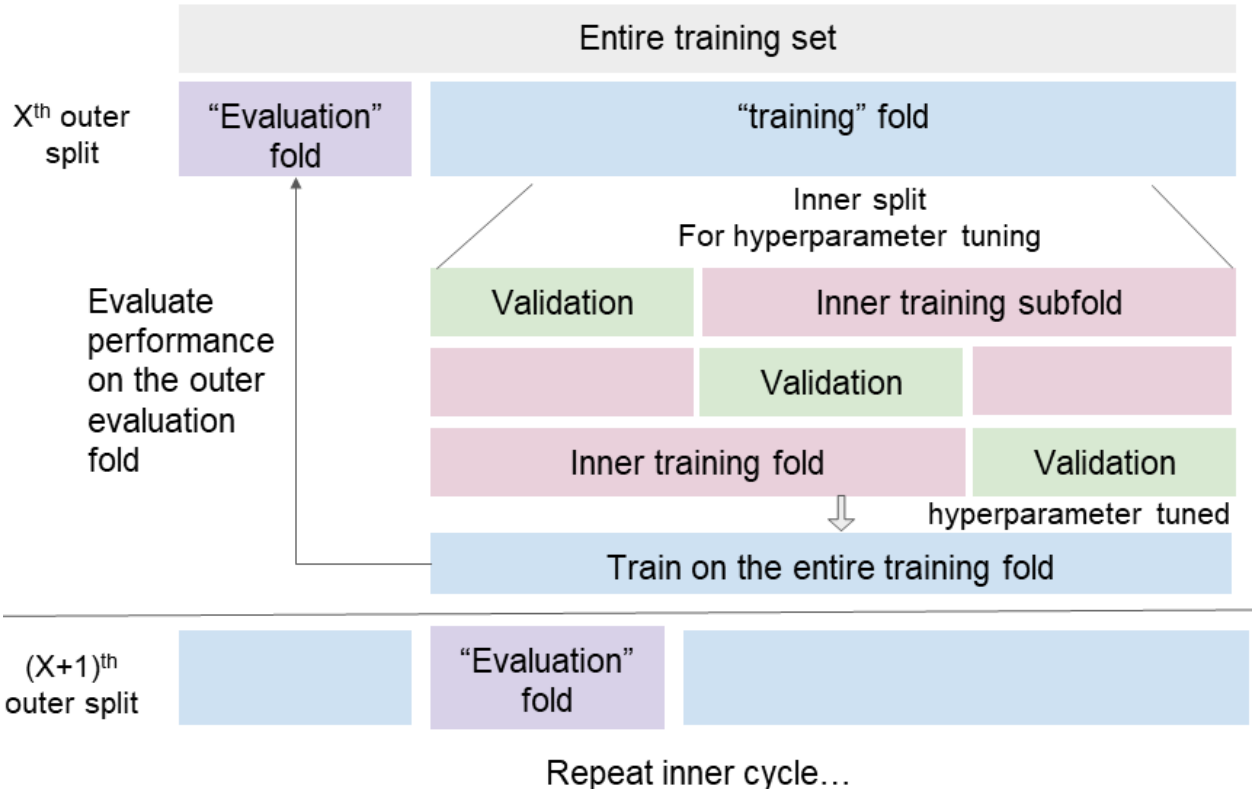



Figure 8: Schematic diagram showing the nested cross-validation scheme to tune the hyper-parameters and select the best model from the candidates for the forward design step. The outer loop of the nested cross-validation scheme divides the entire training set into an evaluation fold and a training fold. In the inner loop, the hyper-parameters for each model are tuned using the training fold, which is further sub-divided into an inner training subfold and a validation subfold. Once the best hyper-parameters are determined, each model is trained on the complete training fold and compared with the evaluation fold. This step is repeated for all splits of the entire training set. The best performing model is trained on the complete dataset.

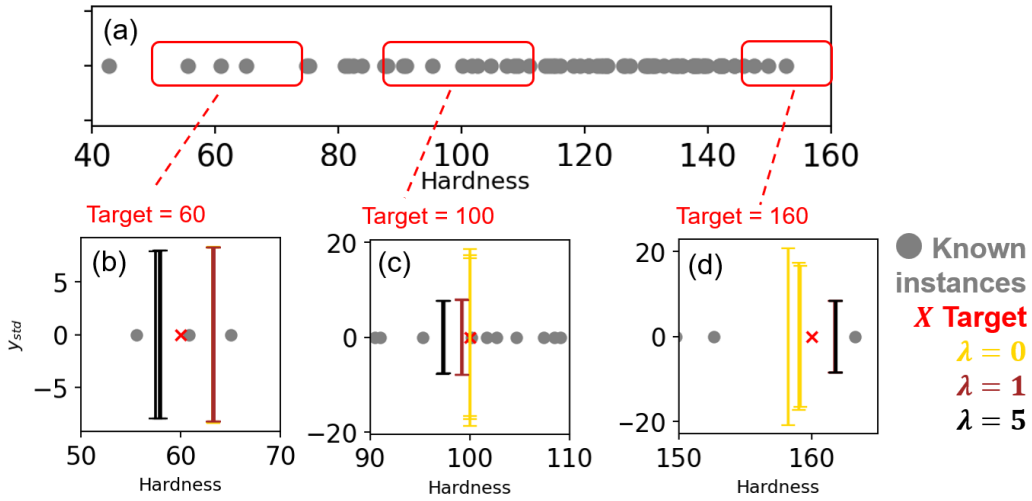
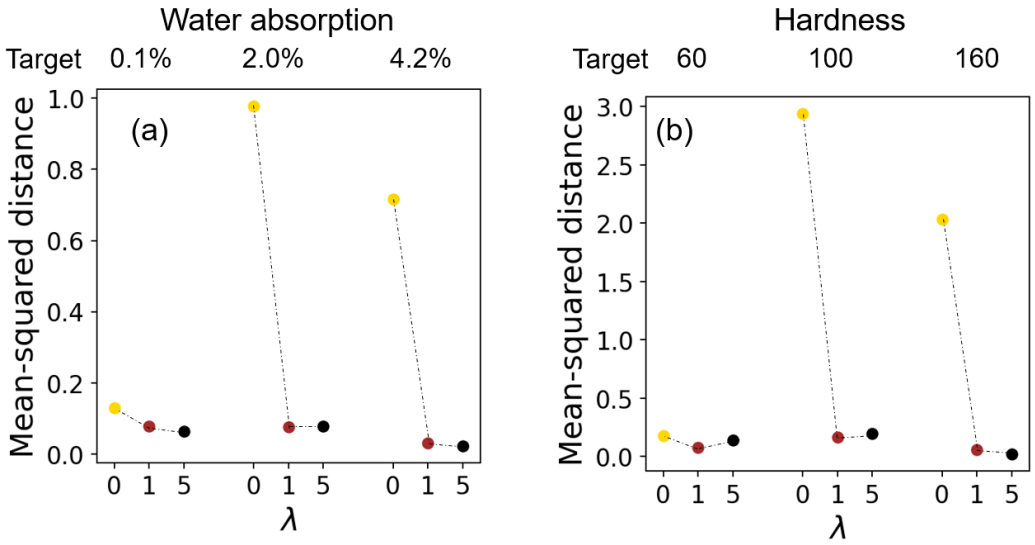



Figure 9: Results of single-target optimization for hardness. (a) Distribution of all hardness values of the training instances, (b-d) Predictions for single-target particle swarm optimization with hardness target = (b) 60, (c) 100, (d) 160. For each (target,$\lambda$) three repeated runs are performed and plotted with three distinct error bars (overlapping when not distinguishableable on the plot). In (b-d), target value is marked by red cross, predicted water absorption values of "best" found formulation are represented by the x-axis values of the colored error bars, and the standard deviation of the predicted distribution of water absorption is represented by error bar along the y-axis.




$$Mean - squared\ distance = \frac{\sum_{i=1}^{N}\sum_{j=i+1}^{N}\left|\boldsymbol{x}_{\boldsymbol{standardized,i}} - \boldsymbol{x}_{\boldsymbol{standardized,j}}\right|^2}{N(N-1)/2}$$

where N = 5 is the total number of replicated searches done for each (target, $\lambda$), and $x_{standardized}, i$ is the optimized feature vector from the i-th replicated search, post-standardization using the same standardizer for the forward model

Figure 10: Mean-squared distance between predicted "best" blend formulation between five independent replicates of same (target, $\lambda$) for single-target particle swarm optimization with (a) water absorption and (b) hardness as target properties