Untangling How Machines 'Learn' Perovskite Crystallization Chemistry

Ian M. Pendleton[1] 0000-0002-4394-3223 ipendlet@umich.edu

Mary K. Caucci[2] 0000-0003-4306-5114 mcaucci@fordham.edu

Aaron Dharna[3] 0000-0002-6261-242X adharna@fordham.edu

Michael Tynes[2,3] 0000-0002-5007-1056 mtynes@fordham.edu

Mansoor Ani Najeeb[1] 0000-0002-8258-0613 maninajeeb@haverford.edu

Zhi Li[4] 0000-0002-8799-0283 zhili@lbl.gov

Emory M. Chan[4] 0000-0002-5655-0146 emchan@lbl.gov

Alexander J. Norquist[1] 0000-0003-1040-5880 anorquis@haverford.edu

Joshua Schrier[2]* 0000-0002-2071-1657 jschrier@fordham.edu

[1] Department of Chemistry, Haverford College, 370 Lancaster Avenue, Haverford, Pennsylvania 19041, USA
[2] Department of Chemistry, Fordham University, 441 E. Fordham Road, The Bronx, New York, 10458, USA
[3] Department of Computer and Information Science, Fordham University, 441 E. Fordham Road, The Bronx, New York, 10458, USA
[4] Molecular Foundry, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, California, 94720, USA

CRediT - Contributor Roles

Ian M. Pendleton: Conceptualization, Data Curation, Investigation, Formal Analysis, Methodology, Software, Visualization, Writing – original draft, Writing - review & editing

Mary K. Caucci: Data Curation, Formal Analysis, Investigation, Visualization

Aaron Dharna: Software, Investigation

Michael Tynes: Software, Investigation, Writing - review and editing

Mansoor Ani Najeeb: Investigation, Validation

Zhi Li: Investigation, Validation

Emory M. Chan: Supervision, Resources, Writing - review & editing

Alexander J. Norquist: Funding acquisition, Resources, Writing - review & editing

Joshua Schrier: Conceptualization, Formal Analysis, Funding acquisition, Project administration, Supervision, Writing – original draft, Writing - review & editing

Table of Contents

Summary of Electronic Supplementary Files

We have included a number of files along with the supporting information. These are stored at DOI referenced links for future use.

TACC (Texas Advanced Computing Center):

NIST MDF / DLHUB:

If accessed through one of the above sources the file will be named '4 - SolVSolUD_DensityPub_SI.zip'

Included in this content:

- **0045.perovksitedata.csv** - main dataset used in this article. A more detailed description can be found in the "dataset overview" section below
- **Chemical Inventory.xlsx** - the hand curated file of all chemicals used in the construction of the perovskite dataset. This file includes identifiers, chemical properties, and other information.
- **ExcessMolarVolumeData.xlsx** - record of experimental data, computations, and final dataset used in the generation of the excess molar volume plots.
- **MLModelMetrics.xlsx** - all of the ML metrics organized in one place (excludes reactant set specific breakdown, see ML_Logs.zip for those files).
- **OrganoammoniumDensityDataset.xlsx** - complete set of the data used to generate the density values. Example calculations included.
- **SolutionVolumeDataset** - complete set of 219 solutions in the perovskite dataset. Tabs include the automatically generated reagent information from ESCALATE, hand curated reagent information from early runs, and the generation of the dataset used in the creation of Figure 5.
- **error_auditing.zip** - code and historical datasets used for reporting the dataset auditing.
- "**AllCode.zip**" which contains:
    - **model_matchup_main_20191231.py** - python pipeline used to generate all of the ML runs associated with the article. More detailed instructions on the operation of this code is included in the "ML Code" Section below. This file is also hosted on
        - GIT: https://github.com/ipendlet/MLScripts/blob/master/model_matchup_main_20191231.py
        - DLHUB:

- **VmE_CurveFitandPlot.py** - python code for generating the third order polynomial fit to the V_m^E vs mole fraction of FAH included in the main text. Requires the 'MolFractionResults.csv' to function (also included).
- **Calculation_Vm_Ve_CURVEFITTING.nb** - mathematica code for generating the third order polynomial fit to the V_m^E vs mole fraction of FAH included in the main text.
- **Covariance_Analysis.py** - python code for ingesting and plotting the covariance of features and volumes in the perovskite dataset. Includes renaming dictionaries used for the publication.
- **FeatureComparison_Plotting.py** - python code for reading in and plotting features for the 'GBT' and 'OHGBT' folders in this directory. The code parses the contents of these folders and generates feature comparison metrics used for Figure 9 and the associated Figure S8. Some assembly required.
- **Requirements.txt** - all of the packages used in the generation of this paper
- **0045.perovskitedata.csv** - the main dataset described throughout the article. This file is required to run some of the code and is therefore kept near the code.

- "**ML_Logs.zip**" which contains:
  - A folder describing every model generated for this article. In each folder there are a number of files:
    - **Features_named_important.csv** and **features_value_importance.csv** - these files are linked together and describe the weighted feature contributions from features (only present for GBT models)
    - **AnalysisLog.txt** - Log file of the run including all options, data curation and model training summaries
    - **LeaveOneOut_Summary.csv** - Results of the leave-one-reactant set-out studies on the model (if performed)
    - **LOOModelInfo.txt** - Hyperparameter information for each model in the study (associated with the given dataset, sometimes includes duplicate runs).
    - **STTSModelInfo.txt** - Hyperparameter information for each model in the study (associated with the given dataset, sometimes includes duplicate runs).
    - **StandardTestTrain_Summary.csv** - Results of the 6 fold cross validation ML performance (for the hold out case)

- **LeaveOneOut_FullDataset_ByAmine.csv** - Results of the leave-one-reactant set-out studies performed on the full dataset (all experiments) specified by reactant set (delineated by the amine)
- **LeaveOneOut_StratifiedData_ByAmine.csv** - Results of the leave-one-reactant set-out studies performed on a random stratified sample (96 random experiments) specified by reactant set (delineated by the amine)
- **model_matchup_main_*.py** - code used to generate all of the runs contained in a particular folder. The code is exactly what was used at run time to generate a given dataset (requires 0045.perovskitedata.csv file to run).

Detailed Experiment Overview

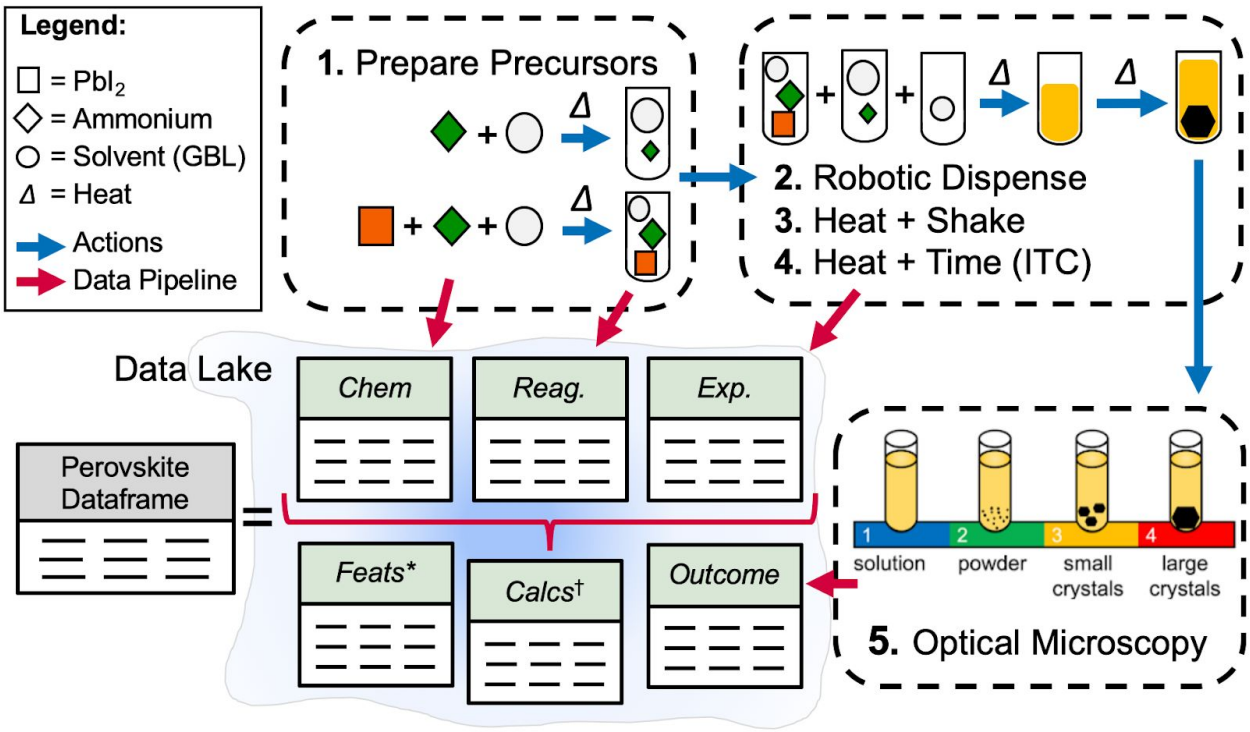

Figure S1. Overview of experimental workflow and data capture for exploration of perovskite process space. *Feats include physicochemical descriptors collected from Chemaxon and RDkit.

The ITC workflow consists of: human preparation of precursor solutions, robotic dispensing of the solutions, a sequence of heating and shaking steps followed by high temperature heating for an extended period of time, and finally evaluation via optical microscopy. Experiments are performed 96-well plates, where the intended reaction conditions and precursor solutions for a given plate are identical. Optical microscopy is used to evaluate the size and quality of the crystals and are reported as classification values (target data for ML prediction): 1 – clear solution, 2 – amorphous powder, 3 – small crystals, 4 – large single-crystals.

While constrained to the ITC workflow, a 'reactant set' can be defined as the accessible range of experimental conditions (shake rates, temperatures, times) and chemical concentrations of the reactants. For an experiment there are four precursor solutions that define the limits of chemical concentrations: 1) γ-butyrolactone (GBL) as the solvent 2) ammonium iodide, lead iodide, and GBL 3) ammonium iodide and GBL 4) formic acid (FAH). Variance within the dataset originates primarily from precursor

preparation (i.e. measured mass and volume of multiple chemical components) and subsequent robotic dispensing of precursor solutions (i.e. dispense volumes) into the 96 well plates. Importantly, while the target volume for an experiment is relatively constant (500-750 µL for an experiment) the dispense volume of precursors can vary significantly between experiments; for example, the dispense volume of FAH ranges from 0 µL to greater than 300 µL. In the final steps of ESCALATE processing the precursor preparation data are used in conjunction with robotic dispense volumes to calculate the final concentrations of each chemical in a given experiment.

Density Calculations

The resulting values of ρ_a, a single value for each combination of organoammonium and halide, were then used in combination with equation (2) to obtain a corrected bulk density ρ_b (i.e., $\rho_{b_{combined}}$, ρ_{b_I}, $\rho_{b_{Br}}$, $\rho_{b_{Cl}}$) from the molecular density. The values for each ρ_b are reported in Table S1 and Figure S2.

Table S1. Linear regression and error metrics for fit between experimental densities and computed densities for organoammonium salts. RMSE is root mean squared error, MAE is mean absolute error.

	Iodide	**Bromide**	**Chloride**	**Combined**
	ρ_{b_I}	$\rho_{b_{Br}}$	$\rho_{b_{Cl}}$	$\rho_{b_{Comb.}}$
Slope (m)	1.02	0.96	0.93	0.83
Intercept (b)	-0.76	-0.52	-0.37	-0.24
R^2	0.95	0.95	0.80	0.96
RMSE (g/mL)	0.07	0.05	0.06	0.07
MAE (g/mL)	0.06	0.05	0.06	0.08

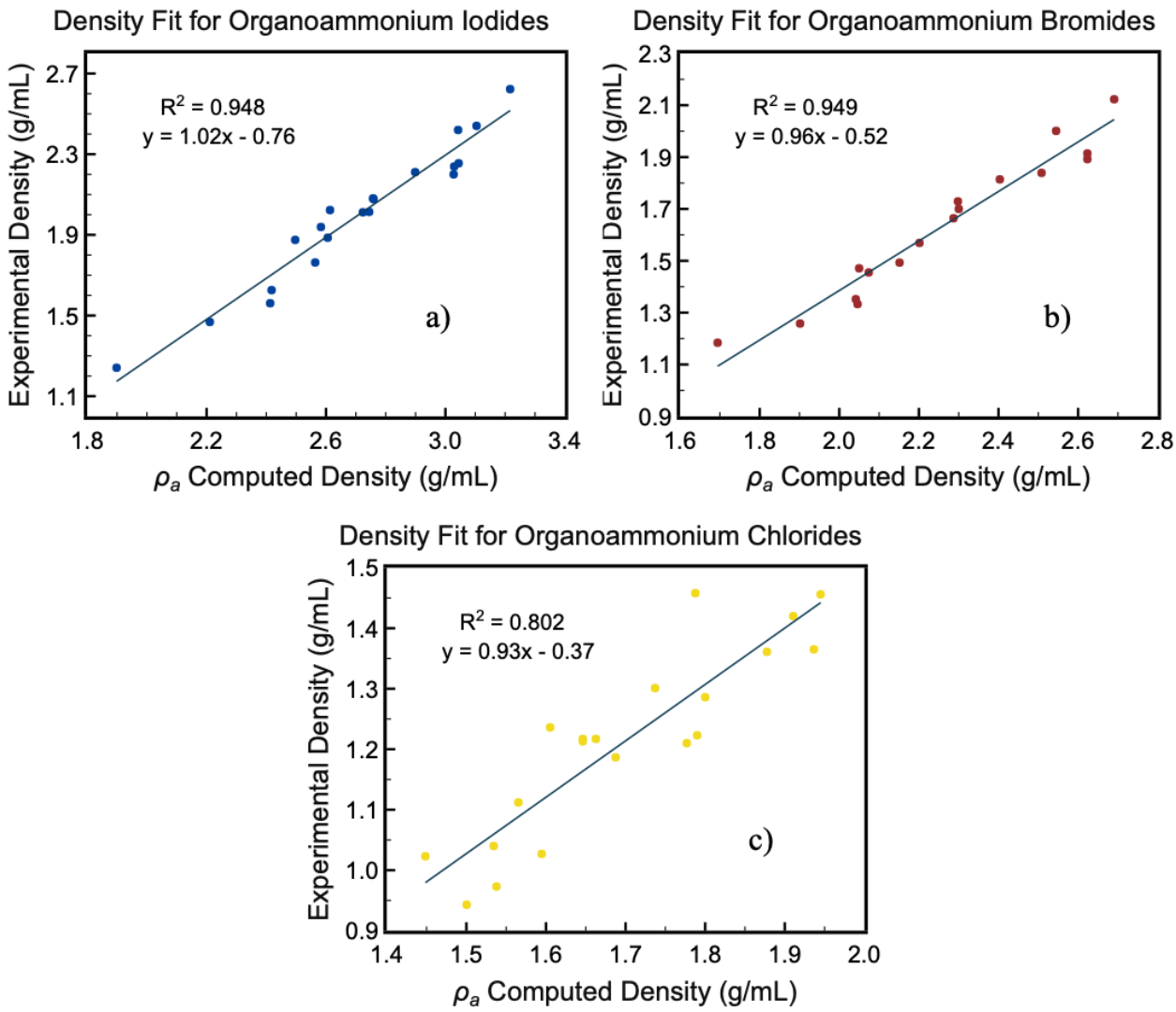

Figure S2. Experimental density values from cambridge structural database (CSD) versus computed densities for (a) organoammonium iodide (b) organoammonium bromide and (c) organoammonium chloride

Table S2 and Figure S3 show the comparison of volumes generated from the SolV approximation, $SolUD_{Mol}$ (SolUD using the uncorrected molecular densities), and the SoLUD model compared to the observed solution volume. This comparison is an extension of Figure 5 included in the main text.

Table S2. Linear regression functions and model fit metrics comparing observed reagent volumes to volumes derived from SolV, $SolUD_{Mol}$, and SolUD solution volume models. MSE is mean squared error and MAE is mean absolute error.

	SolV	$SolUD_{Mol}$	SolUD
Slope (*m*)	1.3	1.1	1.0
Intercept (*b*)	1.6	0.5	0.2
R^2	1.0	1.0	1.0
RMSE (mL)	3.1	2.0	2.0
MAE (mL)	2.2	1.1	0.9

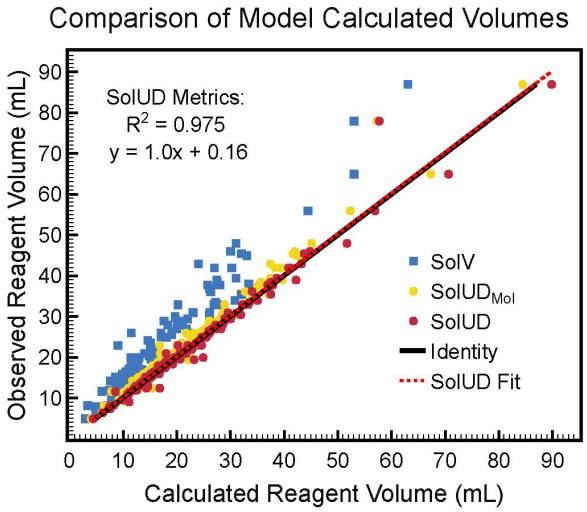

Figure S3. Comparison of calculated reagent volumes to observed volumes with the metrics for the best fit line for the SolUD model.

Auditing Calculations

The following section details the derivation of FAH and GBL mixing error. The percent volume underestimation from omitting non-ideal mixing can be calculated as the sum of the products of partial mole fraction excess molar volumes

$$corrected\ binary\ solution\ volume\ (mL) = \sum_i x_i \left(V^E_{m(x_i)} + \left(\frac{mw_i}{\rho_i} \right) \right)$$

Where: x_i is the mole fraction of component i, $V^E_{m(x_i)}$ is the excess molar volume at the given mole fraction, mw_i is the molecular weight of component i and ρ_i is the density of component i. In the case of GBL and FAH at the maximum possible deviation from ideality, is when the first derivative of the excess molar volume versus mole fraction FAH plot (Figure 4) is equal to zero: mole fraction = 0.36 FAH : 0.64 GBL, V^E_m = 1.33 mL·mol^{-1}. We can calculate the expected behavior of the solution based on the value of V^E_m:

$$volume\ (mL) = V_f = \left(0.36 * \left(1.33\ \frac{mL}{mol} + \frac{46.03\ \frac{g}{mol}}{1.22\ \frac{g}{mL}}\right)\right) + \left(0.64 * \left(1.33\ \frac{mL}{mol} + \frac{86.09\ \frac{g}{mol}}{1.12\ \frac{g}{mL}}\right)\right) = 65.818$$

And the value assuming ideal solution behavior:

$$volume\ from\ ideal\ solution\ behavior\ (mL) = V_i = \left(0.36 * \left(\frac{46.03\ \frac{g}{mol}}{1.22\ \frac{g}{mL}}\right)\right) + \left(0.64 * \left(\frac{86.09\ \frac{g}{mol}}{1.12\ \frac{g}{mL}}\right)\right) = 64.489$$

The percent error due to non-ideality (underestimation) can be calculated as follows :

$$Maximum\ Percent\ Error = \frac{(V_i - V_f)}{V_f} * 100 = 2.02\%$$

The overestimation of concentration can be derived from the error expression in terms of volumes:

$$\%\ overestimation = \frac{\left|\frac{m}{V_i} - \frac{m}{V_f}\right|}{\left|\frac{m}{V_f}\right|} * 100 = \left(\left|\frac{V_f}{V_i} - 1\right|\right) * 100 = 2.06\%$$

Where m is a placeholder for some amount of solute in a given stock solution.

The difference in SolV and SolUD volume estimates can be used to calculate the "error" associated with the SolV model assuming the SolUD model as the ground truth. Overestimation of concentration of the SolV method compared to SolUD can be derived for each reagent (applied via python code included with error_auditing.zip). The code used for calculating the maximum, minimum, standard deviation, and errors

for comparing SolV and SolUD models is included as a python script named, "2 - SolutionVolsComparisonPlots.py" in the supporting information documents.

$$error = \frac{M_{SolV} - M_{SolUD}}{M_{SolUD}} * 100$$

Inorganic Solutions (PbI_2 Containing Solutions)

RunUID with Max Error: 2019-08-02T14_45_54.346271+00_00_LBL_E9

Max Error Percent: 71.6%

RunUID with Min Error: 2019-04-22T22_24_16.919740+00_00_LBL_A7

Min Error Percent: 4.71%

Mean Error Percent: 34.2% ± 15.7%

Organic Solutions (Solutions without PbI_2)

RunUID with Max Error: 2019-03-12T15_22_17.125738+00_00_LBL_C9

Max Error Percent: 126.7%

RunUID with Min Error: 2019-04-22T22_24_16.919740+00_00_LBL_A7

Min Error Percent: 4.71%

Mean Error Percent: 41.0% ± 16.7%

Dataset Auditing Deep Dive

Using historical datasets, curated prior to SolUD and organized via a version-controlled repository, we can follow an audit trail of the ongoing experimental campaign. Validation steps underpin the dissemination of robust datasets from high-throughput campaigns. Reliable calculation of volume provides a direct validation mechanism for human input; comparison of the SolUD derived volumes against the observed volumes can reveal deviations indicative of data entry error.

The improved accuracy of the SolUD method allows for comparison against the limited examples where we had access to measured reagent compositions during the preparation. We used these recorded values (when available) to compare against the SolUD concentrations. Furthermore, we were able to use the updated concentration values to generate accurate convex hulls of accessible process space for each amine. Data entry errors were then able to be identified when the final concentration of the bounding box exceeded the boundary conditions of the convex hull of a given process space (Figure SI4). Four runs with this type of error were identified:

- 2019-02-21T17_06_08.308424+00_00_LBL (reagent 3 value excluded from comparison)
- 2019 -04-04T14_21_13.465936+00_00_LBL (identified as outlier, no action taken, no lab evidence)
- 2018-10-31T17_30_05.725585+00_00_LBL
- 2019-02-21T16_43_20.274786+00_00_LBL

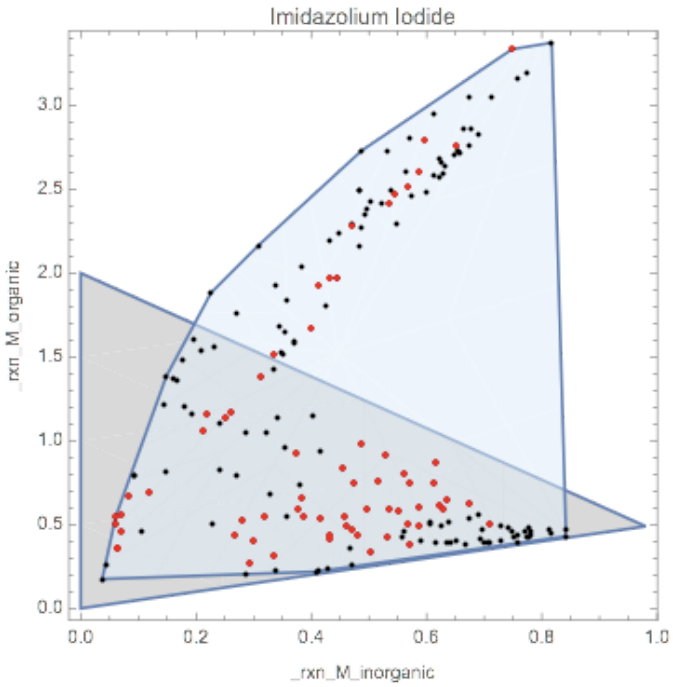

Figure S4. Example of the repercussions of a data entry error. The reported sampled space (blue) exceeds the accessible region of process space (grey) determined from SolUD concentration values.

Another form of data entry validation was implemented based on the SolUD volume models. Reagent preparations that exceeded 10% error from the expected solution volume calculated by SolUD compared to the recorded solution volume were flagged and inspected manually. In some cases, the laboratory notebooks confirmed the data entry and no errors were identified during the run. The following runs were affected:

2018-12-11T15_45_11.271370+00_00_LBL,
2019-04-10T15_53_18.860962+00_00_LBL,
2019-01-21T16_00_00.000000+00_00_LBL,
2019-04-22T22_24_16.919740+00_00_LBL.

The eight plates of reactions identified as data entry errors were resolved by consulting with laboratory notebooks (written records kept by laboratory staff performing the experiments) and replacing the incorrect values (solvent volumes and solute masses) with the correct values from notes. In some cases, there was no written copy to check against and with no obvious means to confirm the existence of an error beyond comparison to the SolUD expected results. In the limited examples where the value was

outside the range expected by SolUD but no paper copy existed, the values in the dataset were assumed to be true. An older dataset, curated prior to fixing the aforementioned errors is included in the supporting information for posterity. The relevant historical CSV files are included in the error_auditing.zip file, and are titled: ComparisonDatasets/0034.perovskitedata.csv and ComparisonDataset/0028.perovskitedata.csv. These files contain errors remedied by work demonstrated in this article. These historic datasets should only be used as a reference for data auditing.

Machine Learning

Code Instructions

The modeling of the various density calculations was performed using Scikit-learn (cite). The final version of code used for modeling comparisons has been included in the supporting information as 'model_matchup_main_20191231.py'. This code can be executed after installing the appropriate libraries. To run the code:

1. Install python 3 from https://www.python.org/downloads/

2. Install the necessary environment using either anaconda or as the default on the OS (the OS will be assumed to be linux or MacOS for simplicity, though windows install is most easily handled through the anaconda environment)

`pip install -r requirments.txt` (requirements.txt is found with the model output and model_matchup_main_**.py code)

3. Ensure that the desired options are set and that 0045.perovksite.csv is included with the code.

4. Edit the model_matchup_main.py to fit with the desired options for reproducing the results (or use the specific instantiation of the `model_matchup_main_**.py` code included with ML_Log.zip) and then execute the code:

$ python3 model_matchup_main_.py**

Where ** indicates the tag in the specific version of the main script.

5. Computational reproducibility can be achieved through inspection / rerunning of the code with the described options. The relevant parameters and control toggles for running models can be found at the end of the `model_matchup_main.py` file. Additional information and descriptions are included in the SI section "model development" below.

6. Workup can be performed by adding all generated run folders (names of individual algorithm-preprocessing) to name "densityconc_done" and executing `python Extract_AmineSpecific_AND_GeneralModel_MetricsV3.py` in the top-level directory containing both the folder and script. The script will generate files in each folder describing the

overall performance of each run. The necessary structure of the resulting runs is demonstrated in the ML_Logs.zip file.

7. Plotting functions require appropriate names for the files (since these are user specified there is no way to predict what the folders will be named). Examples of folder naming and the associated workup scripts are included in the attached zip file.

Though the code can be run in the current state some slight modifications are necessary to reproduce specific instances of the model comparison. Due to the size and scope of the comparison, each model and the subsequent analysis has to be performed independently. The code has not been optimized and multiple iterations will be required to regenerate all of the files. One should note that the code was designed to present 1:1 comparisons of different models for testing. As such, functions in the code often take two datasets (in this case two sub-samples of a given dataset) as input to build models using the algorithm.

For each of the data samples the models are selected through GridsearchCV. The flexibility and search range of the GridSearchCV are defined as a dictionary in the __main__ function and can be changed depending on the time allotment, user constraints and dataset. Standardization, normalization can also be applied prior to running the datasets.

In the ` model_matchup_main.py` code there are instances of inconsistencies in the terminology. Table SI3 below provides a mapping between the language used in the main text and that used in the code. Where possible, comments were added directly to the code to improve readability.

Table S3. Naming scheme mapping used in code and for folders generated using code to the language used in the main text.

Dataset in code	Name in Paper
data0.raw1	SolV + Chem
data1.raw1	SolUD + Chem
data0.raw2	SolV + Exp
data1.raw2	SolUD + Exp
data0.raw3	SolV + Reag
data1.raw3	SolUD + Reag
data0	SolV
data1	SolUD
data1.pM	SolUD (prop.)
noConc.raw1	Chem
noConc.raw2	Exp
noConc.raw3	Reagent
noConc	Feats* Only

Data Preprocessing

The performance of models was also assessed using a series of normalization and standardization tests. The normalization and standardization was developed on the training data and the fit applied to the testing data for each split of the cross-validation. The three tiers of preprocessing were as follows:

1. Normalization (norm) and standardization (std) was applied to columns with the *_feat_** prefix to compare against the raw dataset. The preprocessing of these features was performed on runs indicated with _norm0 or _std0 suffixes.
2. Additionally, normalization and standardization were applied to the *_rxn_** prefixes (including both SolV and SolUD columns in these cases). The models built on these preprocessed features are represented by _norm1 and _std1 respectively.
3. Finally, all remaining features were included in the last tier of normalization and standardization. These models are denoted _norm2 and _std2

To further investigate the effects of physicochemical features on the performance of models we also developed a comparison of model performance with all *_feats_** except the 'Avg Surface Area' removed and the same feature represented through one-hot-encoding. This model is denoted with a _OHE (one-hot-encoding) suffix.

Model Development

The modifications that are necessary are all available and clearly outlined in the code under the standard main executable function:

` if __name__ == '__main__'`

The options used in the generation of each model are delineated within the ` model_matchup_main_**.py` code. Hyperparameter Tuning was performed for each model using GridSearchCV. A copy of the dictionaries used to define the constraints on GridSearchCV are included in the model_matchup_main code. For kNN memorization benchmark the number of nearest neighbors was constrained to 1 while in other kNN the number of neighbors was used as a variable in grid search. All amines considered (in both the STTS and the LOO) required at least one demonstrated success within the given space. If no successes are observed for a particular amine the rest of the data tied to that amine are omitted from test-train splits. The omission of data is consistent across all permutations of the dataset.

Accessing Model Details

The optimized configuration for each model pipeline is reported below and the details associated with the hyperparameter tuning of a specific model can be found in the zip file associated with this document. The results are included in the '4 Crystallization Prediction by Machine Learning' folder.

GBT Example

Additional documentation can be found on Scikit-learn: Machine Learning in Python (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensemble.GradientBoostingClassifier).

Notes on baselines: Not all of the data was shuffled during the deep shuffle. Some of the columns are codependent or used in calculation of derived features. For example, shuffling the final concentrations in a given experiment (_raw_M_<inchikey>_final) as well as the maximum possible concentration of a reagent (_raw_reagent_conc_<inchikey>) would prohibit implementation of the 'proportional concentration' calculations. The mapping between the files in ML_logs.zip and the naming scheme used in the publication can be found in Table S3 above.

Looking up details for the best performing STTS GBT model:

GBT algorithm, the *SolV + Chem* data sample, and standardized features (i.e.std0), Full dataset not sampled(i.e. stratified=False)

can be done as follows:

1. Unzip ML_Logs.zip, navigate to ML_Logs/gbt_std0
2. Hyperparameters for the model: Open STTSModelInfo.txt , go to line starting with the data sample name (SolV + Chem = data0.raw1). The output is included after this walkthrough.
3. Note: that stratify=False means all the data was used in the construction of the model while stratify=True indicates 96 random samples from each amine were used for the training
4. Note: The same model may appear multiple times in the file. This is expected behavior as the original code was written to compare two models, multiple instances of a single pipeline may have been called, but the results should be identical (random state is set to 42 for all runs to ensure computational reproducibility).
5. This dictionary contains all of the necessary information to fully reproduce the model in the same version of Scikit-Learn used in this article.

Example Hyperparameter output for GBT algorithm, the *SolV + Chem* data sample, and 'standardized features-std0':

```
n_jobs=None, remainder='passthrough',
                    sparse_threshold=0.3,
                    transformer_weights=None,
                    transformers=[('standardscaler',
                                   StandardScaler(copy=True,
```

with_mean=True,

with_std=True),

['_feat_fr_piperdine',

'_feat_RingAtomCount',

'_feat_VanderWaalsSurfaceArea',

'_feat_msareaASAp',

'_feat_molsurfaceareaVDWp',

'...

learning_rate=0.1, loss='deviance',

max_depth=6, max_features=None,

max_leaf_nodes=None,

min_impurity_decrease=0.0,

min_impurity_split=None,

min_samples_leaf=2,

min_samples_split=6,

min_weight_fraction_leaf=0.0,

n_estimators=100,

n_iter_no_change=None,

presort='auto', random_state=42,

subsample=1.0, tol=0.0001,

validation_fraction=0.1, verbose=0,

warm_start=False))],

verbose=False)

kNN Examples

Additional documentation can be found on Scikit-learn: Machine Learning in Python https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

An identical procedure can be applied to kNN models.

kNN 1

A benchmark method example for 'memorized' learning (what example is closest is sufficient from previous run).

data0. , stratify= True , KNeighborsClassifier(algorithm='ball_tree', leaf_size=30, metric='minkowski', metric_params=None, n_jobs=3, n_neighbors=1, p=2, weights='uniform')

kNN

Best performing kNN referenced in the main article:

kNN, *Chem*, standardized features (i.e.std2), Full dataset not sampled (i.e. stratified=False) , MCC = 0.66 ± 0.05, 109 total features

noConc.raw1 , stratify= False , KNeighborsClassifier(algorithm='ball_tree', leaf_size=30, metric='minkowski', metric_params=None, n_jobs=None, n_neighbors=5, p=2, weights='uniform')

Linear SVC

Additional documentation can be found on Scikit-learn: Machine Learning in Python (https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC)

Additional ML Analysis

Using Matthew's correlation coefficient (MCC). The cross section of samples analyzed by gbt provides further evidence that the representation of the data has only marginal impact on the overall performance of these models.

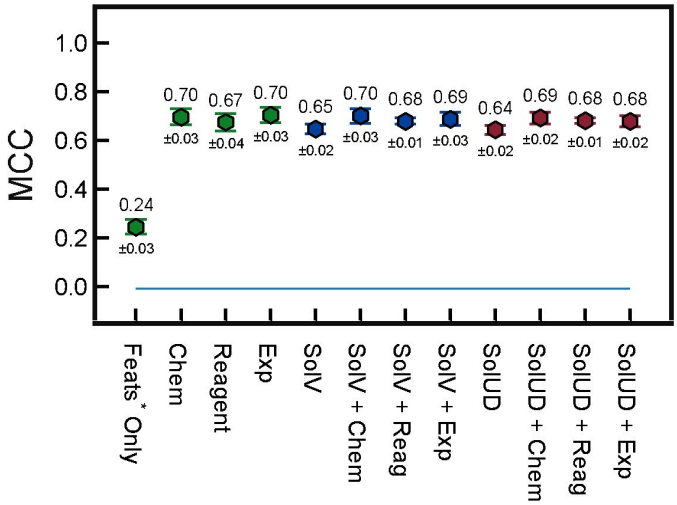

Figure S5. MCC comparison of GBT model performance across all data samples. Error bars and labels describe the standard deviation of performance for held-out data.

The best kNN models perform worse, but similar to the best GBT models.

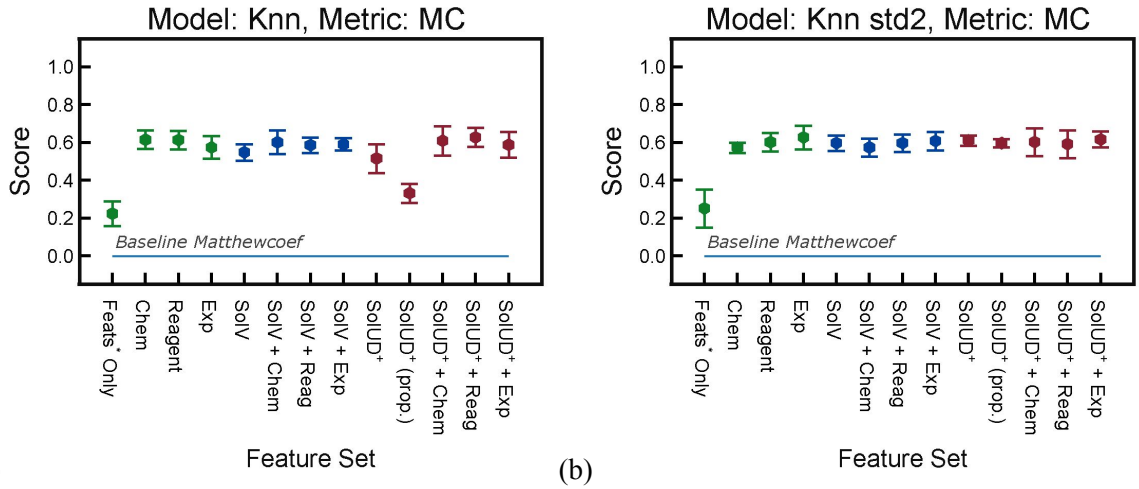

Figure S6. Comparison of kNN model performance for (a) kNN and (b) kNN with all features standardized

Furthermore, once simple standardization techniques are applied to the _feat_ prefixed namespace in the dataset (columns which contain exclusively physicochemical descriptors), the STTS performance changes from each data sample is negligible. This effect can be visualized by comparison of gbt data samples across four preprocessing pipelines with the best performance (Figure S7).

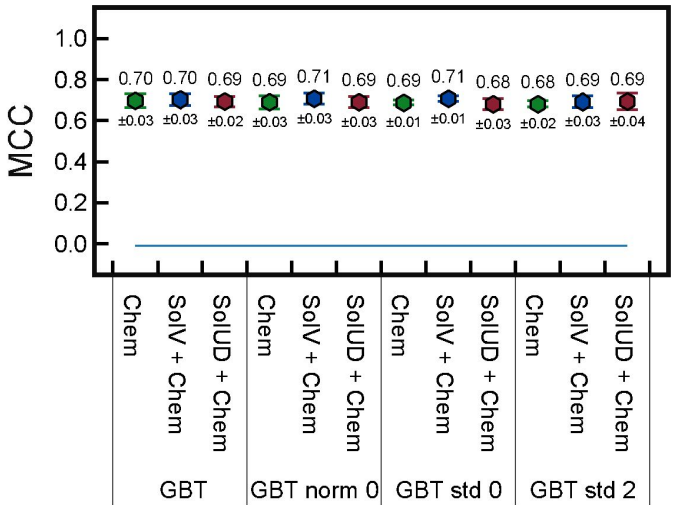

Figure S7. Comparison of top STTS GBT models using various preprocessing regimes.

All the performance metrics for all ML models are included in MLModelMetrics.xlsx

Table S4. Analysis of the feature contributions for model trained using GBT algorithm, *Chem* data sample and no preprocessing. The first five features are equivalent to the values shown in Figure 9 of the main text.

Feature Name	Weighted Contribution Value
Organic Dispense Vol.	1.02965587
Solvent Dispense Vol.	0.89927081
Inorganic Dispense Vol.	0.75879927
Avg Surface Area	0.73557126
Acid Dispense 1	0.44915013
Acid Dispense 2	0.37312617
Organic Dispense (Slot 2)	0.33712649
Inorganic Solution 1- GBL Volume	0.14573481
FAH total volume (Slot 1)	0.14501565
Inorganic Solution 1 - Organoammonium mass	0.12910601
Polar Surface Area	0.12078133
Organic Solution 1 - Organoammonium mass	0.1202994
FAH total volume (Slot 2)	0.12024083
Inorganic Solution - PbI2 mass	0.1057313
Organic Solution 1 - GBL Volume	0.10469305
Inorganic Dispense Vol. 2	0.10452502
GBL Volume (on robot deck)	0.10305632
'_feat_LengthPerpendicularToTheMaxArea'	0.06586922
Organic Solution 2 - Organoammonium mass	0.04894347
Reaction Temperature	0.03992693

Notes on the table above. The Hamilton robot deck supports up to 9 solutions. Most of the experiments were performed using the slots as follows: 1-GBL, 2-Inorganic Solution (Consisting of GBL, organoammonium salt, and lead diiodide), 3-Organic Solution (GBL, organoammonium salt) 6-FAH (Slot 1), 7-FAH (Slot 2). In cases where a second ammonium was used in parallel experiments slow 4 and 5 would be dedicated to the 'inorganic solution 2' and 'organic solution 2', respectively. More information

about the features ‘Polar Surface Area’ and ‘_feat_LengthPerpendicularToTheMaxArea' can be found in the feature description section below.

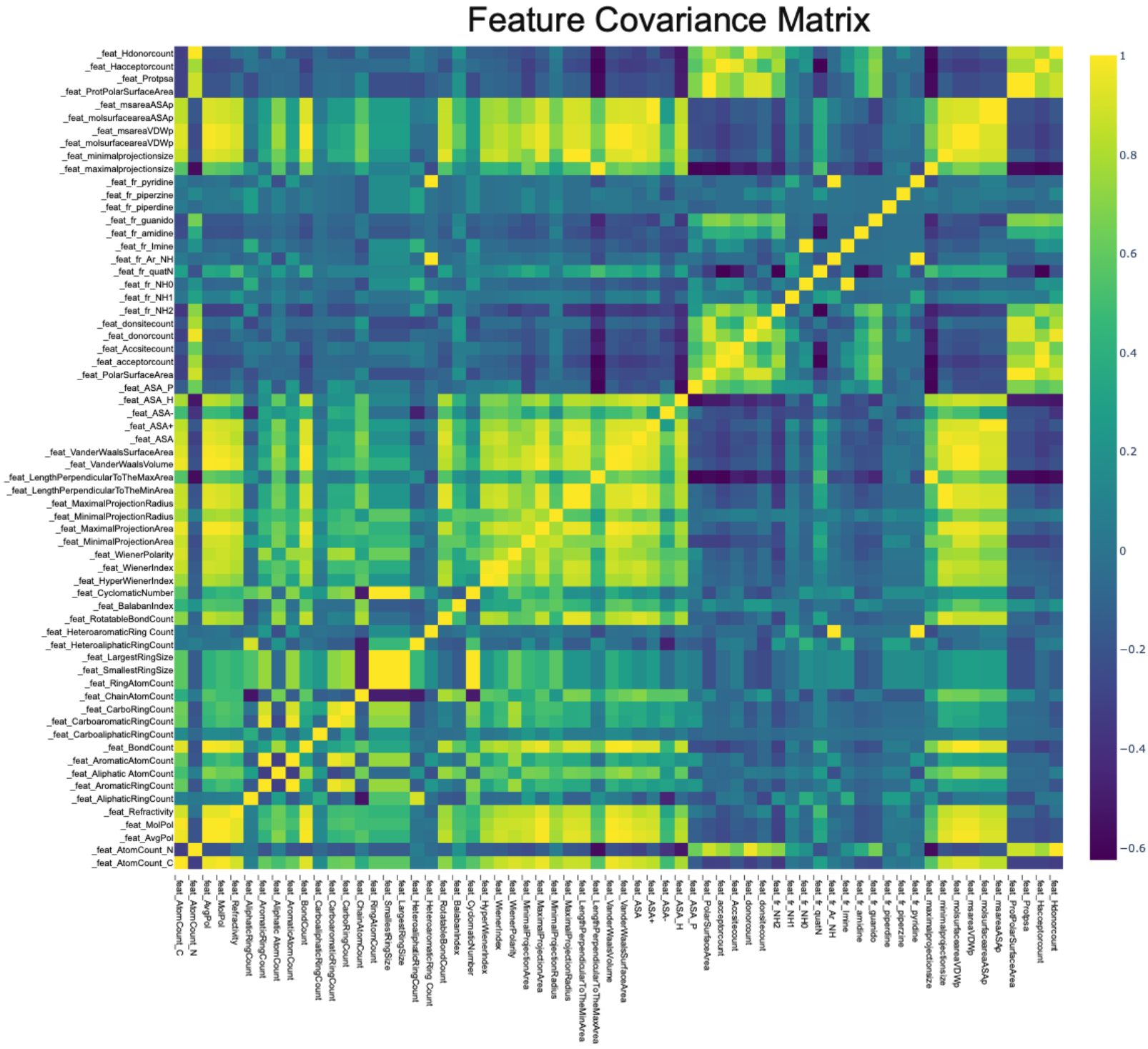

Figure S8. Covariance matrix of physicochemical descriptors included in the perovskite dataset.

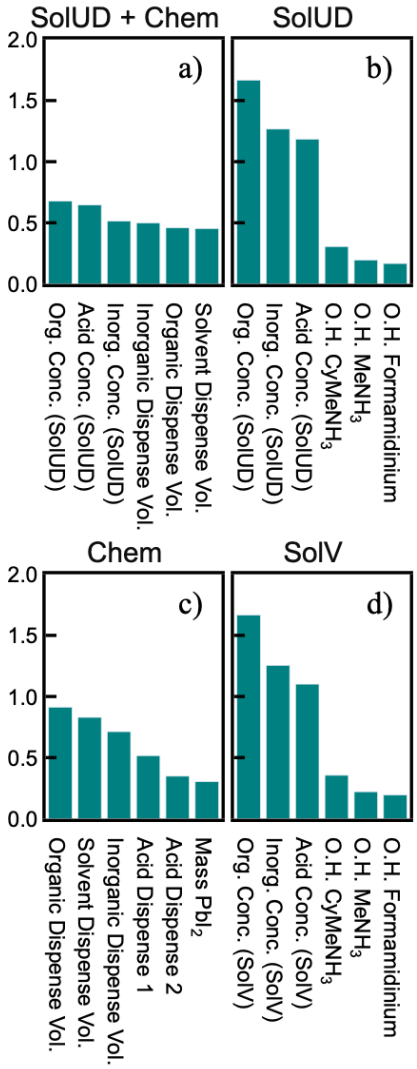

Figure S9. One Hot Encode (OHE) GBT model weighted feature contributions for a) SolUD + Chem b) SolUD c) Chem and d) SolV data samples

Dataset Overview

Explanation Header Prefixes

raw

DO NOT LEARN ON. We strongly recommend not using these unless you know what the header means and how the column of data relates to experimental outcomes. Raw data associated with the experiment that was performed. Combined with the other header prefixes, these columns describe the complete set of all data acquired during an experimental run.

rxn

Recommended set of data to learn on. These descriptors include experimental observables, reaction conditions, and calculated molecular features. More elaborated descriptions can be found below for individual headers.

feat or _calc_

Recommended set of data to learn on

prototype

New features that should be used with caution. These are under development and likely require additional integrity testing. Specific features may require additional process or special handling before consumption by typical ML tools.

out

Target outputs to predict. These are a numeric representation of the manually scored crystal quality of a single reaction rated on a scale of 1 to 4. For the organohalide perovskite chemistry, a “4” indicates that

large single crystals were observed; this is the ideal outcome of the experiment. A "3" means many smaller crystals were observed; still an indicator of potentially useful material, but less desirable than a "4". A "2" indicates that a powder or only a few small crystals were observed. A "1" represents no observable formation of perovskite solid.
If used for binary classification, the outcomes can be changed to Booleans reasonably in two ways:

- 1s and 2s can be rated failures (0) and 3s and 4s can be rated successes (1).
- 1s, 2s, and 3s are rated as failures (0) and 4s only are successes (1)

Ideally, the goal would be to predict only 4s as successes, but it is acceptable to tackle the potentially easier problem of predicting both 3s and 4s as successes first.

Additional Notes

The provided training subset was selected from the total raw data set through a two-step screening process.

Both conditions must be satisfied for data to be pulled from the RAW set to the curated data set:

1. Supplies the model with features we would like to optimize
    - Examples of *good* features: chemical descriptors, mmol of reactant (i.e. not solvent mmol), well temperature, volume of pure chemicals, total volumes of solutions which vary between experiments
    - Examples of *bad* features: Operator name, run date, run id, grams of a chemical in a reagent, reagent preparation data, etc.
2. Captures variance in the current combined set of experiments
3. Proven to be easily implementable or commonly understood
4. Has been implemented successfully for new modeling campaigns without significant struggle

General Nomenclature

A process space is defined as the combination of chemicals bounded by a series of workflow actions which fall within a particular constraint… A state space is the large space defined by combinations of

various similar process spaces. In general process spaces are targeted by exploitation algorithms and are therefore most often related to interpolative analysis. Much of the code used herein describes the STTS as being interpolative models and LOO being extrapolative models.

- Run
    - The top level view of the tray of experiments performed. Signified by a unique identifier "Run_ID". Has properties such as lab, user, date, time, and workflow identification.
- Tray
    - A single set of 96 experiments/reactions which are performed together. THe properties associated with the tray are applied to all of the wells (and thereby the experiments/reactions). The tray is described by a unique "Run" identifier.
- Well
    - The location on the tray where the reaction/experiment is taking place. Some properties of wells vary throughout the tray such as temperature.
- User
    - The human or organization performing the stock solution preparation and executing the job.
- Experiment/Reaction
    - A specific test which is described by the properties of the environment and the reagents/chemicals added to the "well" in which the experiment is taking place.
- Organic
    - In the case of the perovskite chemistry the term "organic" is referring to the amine used in the experiment.
- Inorganic
    - In the case of the perovskite chemistry the term "inorganic" is referring to the metal used to form the perovskite. For these data, inorganic only refers to lead diiodide (PbI_2)
- Reagent
    - A chemical or combination of chemicals which create the solution added by the robot to the well.
- Chemical

    - A compound which can be defined by an InChIkey. Can be of various qualities and purities, but should be primarily defined by the core component molecule. This is the most granular definition of what is added to each experiment and thereby, each well.
- Solvent
    - The chemical used to solvate the organic and inorganic component of the reaction facilitating transfer on the robot.Namespace in the CSV:

Explanation of Features-Descriptors

The following section provides a general description of each of the features used in the current dataset.

rxn → Experimental Features

- _rxn_mmol_inorganic_actual
    - mmol of the inorganic component added to the reaction well.
- _rxn_mmol_organic_actual
    - mmol of the organic component added to the reaction well. Descriptions of the variance in the organic components are found in the _feat_ section of the dataset. Additional descriptions are under development.
- _rxn_mixingtime1S
    - Duration of the first mixing time after the addition of solvent, organic, inorganic and the first addition of formic acid.
- _rxn_mixingtime2S
    - Duration of mixing time (in seconds) after the addition of the second portion of formic acid.
- _rxn_mmol_<string>_final
    - The specific organic component used in the reaction. <string> refers to the string of characters in the namespace of the CSV. This string is the chemical InChIKey. These have been excluded from the current provided dataset. Ideally however, the variance in the amine should be captured in the _feat_ features. For more information on InChIKeys, please see: https://en.wikipedia.org/wiki/International_Chemical_Identifier
- _rxn_reactiontimeS

    - Duration of time the reaction was allowed to sit and for crystals to form after the initial chemical mixing.
- _rxn_stirrateRPM
    - The rate at which the tray was shaken during the mixing time. **Note:** This number is not consistent for all runs in the dataset, but is potentially hazardous to learn on. The runs performed using different shake rates also have other characteristics which distinguish them. Care should be used when drawing conclusions from models which show this as an important feature.
- _rxn_temperatureC
    - The temperature of the reaction in C.

calcs → Calculated Features

Currently only concentration features found under the _rxn_ namespace are calculated internal to ESCALATE. Additional upcoming development is described below.

Proportional Concentration

These features are generated in line with the model_matchup_main_**.py code. These values are generated with the headers '_rxn_proportionalConc_v1-organic', '_rxn_proportionalConc_v1-inorganic', and '_rxn_proportionalConc_v1-acid' when derived from the SolUD model and '_rxn_proportionalConc_organic', '_rxn_proportionalConc_inorganic', and '_rxn_proportionalConc_acid' when derived from the SolV model. These values are not stored in the 0045.perovskite.csv file. The calculation is straight forward:

Species proportional Conc = (species Experiment Conc) / (max species Reagent Conc)

Where the proportional concentration of a given species (organic, inorganic, acid) is determined by the concentration of that species in a given experiment divided by the maximum accessible concentration for that species. The maximum accessible concentration is a measured experimental parameter determined through solubility measurements during workflow development.

feats → Physicochemical Descriptors Overview

If information regarding a particular feature cannot be found in the list below, please refer to the linked documentation.

Feature Contribution Overview

Expert features have been curated by domain experts as a possible alternative to existing physicochemical descriptors from rdkit, chemaxon, openbabel, etc. These features are hand-curated and could possibly incorporate unintentional errors. They are included under the _prototype_ namespace to avoid accidental inclusion into automated data processing pipelines.

This table last updated = Thursday, November 21, 2019

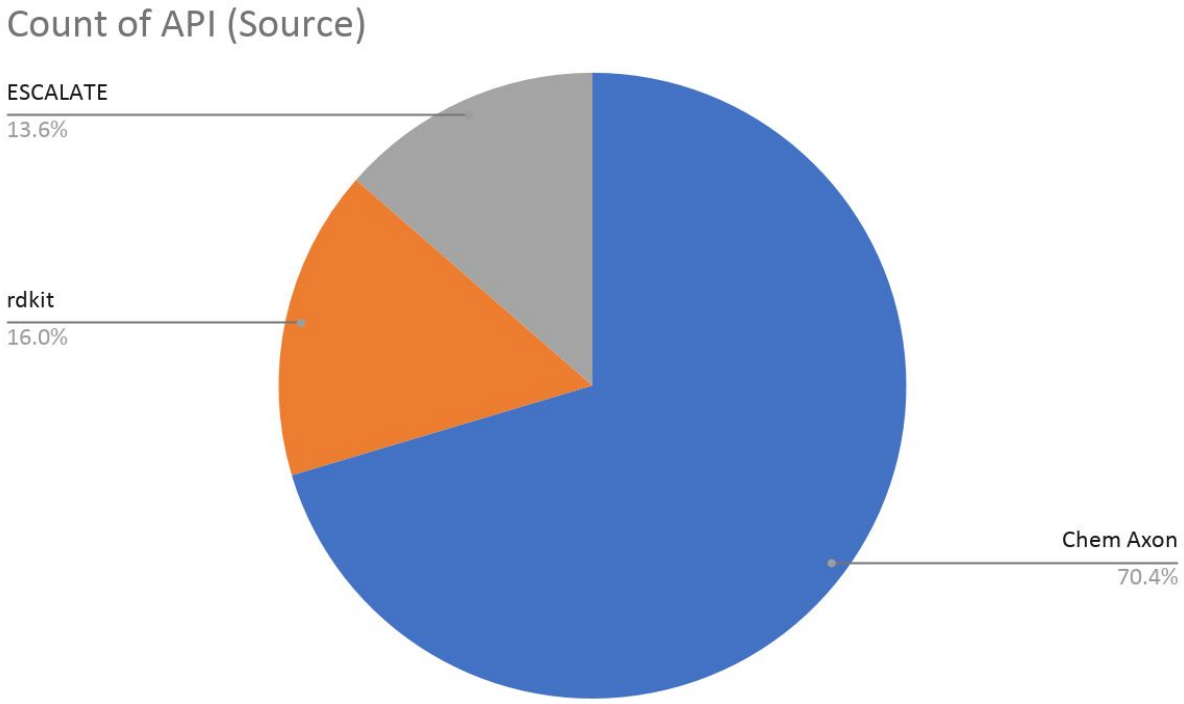

Figure S10. Feature distribution in the perovskite dataset

Detailed Feature Descriptions

ChemAxon

The complete description of the ChemAxon functions can be found here:
https://docs.chemaxon.com/display/docs/cxcalc+calculator+functions

Feature UID	Description	API (Source)	Source Version	Source Function
_feat_acceptorcount	Hydrogen bond acceptor atom count in molecule	ChemAxon: cxcalc	19.24.0	acceptorcount
_feat_Accsitecount	Hydrogen bond acceptor multiplicity in molecule (more details on web)	ChemAxon: cxcalc	19.24.0	acceptorsitecount
_feat_Aliphatic AtomCount	Counts the number of aliphatic atoms in the molecule	ChemAxon: cxcalc	19.24.0	aliphaticatomcount
_feat_AliphaticRingCount	Aliphatic ring count	ChemAxon: cxcalc	19.24.0	aliphaticringcount
_feat_AromaticAtomCount	Aromatic atom count	ChemAxon: cxcalc	19.24.0	aromaticatomcount

_feat_AromaticRingCount	Aromatic ring count	Chem Axon: cxcalc	19.24.0	aromaticringcount
_feat_ASA	solvent accessible surface area calculated using the radius of the solvent (1.4 Å for the water molecule)	Chem Axon: cxcalc	19.24.0	wateraccessiblesurfacearea
_feat_ASA_H	solvent accessible surface area of all hydrophobic (|qi|<0.125) atoms (|qi| is the absolute value of the partial charge of the atom)	Chem Axon: cxcalc	19.24.0	wateraccessiblesurfacearea
_feat_ASA_P	solvent accessible surface area of all polar (|qi|>0.125) atoms (|qi| is the absolute value of the partial charge of the atom)	Chem Axon: cxcalc	19.24.0	wateraccessiblesurfacearea
_feat_ASA-	solvent accessible surface area of all atoms with negative partial charge (strictly less than 0)	Chem Axon: cxcalc	19.24.0	wateraccessiblesurfacearea
_feat_ASA+	solvent accessible surface area of all atoms with positive partial charge (strictly greater than 0)	Chem Axon: cxcalc	19.24.0	wateraccessiblesurfacearea
_feat_AtomCount_C	Number of Carbon atoms in the molecule	Chem Axon: cxcalc	19.24.0	atomcount -z 6
_feat_AtomCount_N	Number of Nitrogen atoms in the molecule	Chem Axon: cxcalc	19.24.0	atomcount -z 7
_feat_AvgPol	Average molecular polarizability calculation	Chem Axon: cxcalc	19.24.0	avgpol

_feat_BalabanIndex	The Balaban index	Chem Axon: cxcalc	19.24.0	balabanindex
_feat_BondCount	Bond count	Chem Axon: cxcalc	19.24.0	bondcount
_feat_CarboaliphaticRingCount	Carboaliphatic ring count	Chem Axon: cxcalc	19.24.0	carboaliphaticringcount
_feat_CarboaromaticRingCount	Carboaromatic ring count	Chem Axon: cxcalc	19.24.0	carboaromaticringcount
_feat_CarboRingCount	Number of rings containing only carbon atoms	Chem Axon: cxcalc	19.24.0	carboringcount
_feat_ChainAtomCount	Number of atoms in aliphatic chains	Chem Axon: cxcalc	19.24.0	chainatomcount
_feat_ChiralCenterCount	The number of tetrahedral stereogenic center atoms	Chem Axon: cxcalc	19.24.0	chiralcentercount
_feat_CyclomaticNumber	The cyclomatic number (complexity of molecule metric)	Chem Axon: cxcalc	19.24.0	cyclomaticnumber

_feat_donorcount	Hydrogen bond donor atom count in molecule	Chem Axon: cxcalc	19.24.0	Donorcount
_feat_donsitecount	Hydrogen bond donor multiplicity in molecule (more details on website)	Chem Axon: cxcalc	19.24.0	donorsitecount
_feat_Hacceptorcount	Hydrogen bond acceptor multiplicity in molecule (at pH 3.0)	Chem Axon: cxcalc	19.24.0	acceptorcount -H 3.0
_feat_Hdonorcount	Hydrogen bond donor atom count in molecule (at pH 3.0)	Chem Axon: cxcalc	19.24.0	donorcount -H 3.0
_feat_HeteroaliphaticRingCount	number of heteroaliphatic rings in molecule	Chem Axon: cxcalc	19.24.0	heteroaliphaticringcount
_feat_HeteroaromaticRing Count	number of heteroaromatic rings in molecule	Chem Axon: cxcalc	19.24.0	heteroaromaticringcount
_feat_HyperWienerIndex	Hyper Wiener index	Chem Axon: cxcalc	19.24.0	hyperwienerindex
_feat_LargestRingSize	Number of atoms in largest ring	Chem Axon: cxcalc	19.24.0	largestringsize
_feat_LengthPerpendicularToTh eMaxArea	Calculates the size of the molecule perpendicular to the maximal projection area surface	Chem Axon: cxcalc	19.24.0	maximalprojectionsize

_feat_LengthPerpendicularToTheMinArea	Calculates the size of the molecule perpendicular to the minimal projection area surface	Chem Axon: cxcalc	19.24.0	Minimalprojectionsize
_feat_MaximalProjectionArea	Calculates the maximal projection area	Chem Axon: cxcalc	19.24.0	maximalprojectionarea
_feat_MaximalProjectionRadius	Calculates the maximal projection radius	Chem Axon: cxcalc	19.24.0	maximalprojectionradius
_feat_maximalprojectionsize	Calculates the size of the molecule perpendicular to the maximal projection area surface	Chem Axon: cxcalc	19.24.0	maximalprojectionsize
_feat_MinimalProjectionArea	Calculates the minimal projection area	Chem Axon: cxcalc	19.24.0	minimalprojectionarea
_feat_MinimalProjectionRadius	Calculates the minimal projection radius	Chem Axon: cxcalc	19.24.0	minimalprojectionradius
_feat_minimalprojectionsize	Calculates the size of the molecule perpendicular to the minimal projection area surface	Chem Axon: cxcalc	19.24.0	minimalprojectionsize
_feat_MolPol	Molecular polarizability calculation	Chem Axon: cxcalc	19.24.0	molpol

_feat_molsurfaceareaASAp	solvent accessible surface area of all atoms with positive partial charge (at pH 3.0)	Chem Axon: cxcalc	19.24.0	molecularsurfacearea -t ASA+ -H 3.0
_feat_molsurfaceareaVDWp	calculates the van der Waals surface of the molecule (at pH 3.0)	Chem Axon: cxcalc	19.24.0	molecularsurfacearea -t vanderwaals -H 3.0
_feat_msareaASAp	Molecular Surface Area calculation of atoms with positive partial charge (at pH 3.0)	Chem Axon: cxcalc	19.24.0	msa -t ASA+ -H 3.0
_feat_msareaVDWp	van der Waals surface calculation of atoms with positive partial charge (at pH 3.0)	Chem Axon: cxcalc	19.24.0	msa -t vanderwaals -H 3.0
_feat_PolarSurfaceArea	Topological Polar Surface Area calculation (2D)	Chem Axon: cxcalc	19.24.0	polarsurfacearea
_feat_ProtPolarSurfaceArea	Topological Polar Surface Area calculation (2D) (at pH 3.0)	Chem Axon: cxcalc	19.24.0	polarsurfacearea -H 3.0
_feat_Protpsa	Topological Polar Surface Area calculation (at pH 3.0)	Chem Axon: cxcalc	19.24.0	psa -H 3.0
_feat_Refractivity	Molecular refractivity calculation (derived from polarizability)	Chem Axon: cxcalc	19.24.0	refractivity
_feat_RingAtomCount	Number of atoms in molecular rings	Chem Axon: cxcalc	19.24.0	Ringatomcount

_feat_RotatableBondCount	Number of rotatable atomic bonds in the molecule(s)	Chem Axon: cxcalc	19.24.0	Rotatablebondcount
_feat_SmallestRingSize	Number of atoms in smallest ring	Chem Axon: cxcalc	19.24.0	smallestringsize
_feat_VanderWaalsSurfaceArea	Van der Waals Surface Area calculation	Chem Axon: cxcalc	19.24.0	vdwsa
_feat_VanderWaalsVolume	Calculates the van der Waals volume of the molecule	Chem Axon: cxcalc	19.24.0	volume
_feat_WienerIndex	Wiener index	Chem Axon: cxcalc	19.24.0	wienerindex
_feat_WienerPolarity	Wiener polarity	Chem Axon: cxcalc	19.24.0	wienerpolarity
_raw_molweight	molecular weight based on given SMILES representation	Chem Axon: cxcalc	19.24.0	mass
_raw_standard_molweight	molecular weight of the standardized smiles string	Chem Axon: cxcalc	19.24.0	mass

Other

The complete description of the rdkit functions can be found here:
http://www.rdkit.org/Python_Docs/rdkit.Chem.Fragments-module.html

Feature UID	Description	API (Source)	Source Version	Source Function
_prototype_ECFP4_hexstring	circular topological fingerprints designed for molecular characterization (advanced - please reach out for help)	Chem Axon	19.24.0	eneratemd c input.smiles -k ECFP -c ecfp_config.xml -2
_raw_smiles_standard	curates a standardized smiles string from input smiles	Chem Axon	19.24.0	standardize
_prototype_ringcountBOOL	Rings present (boolean)	ESCALATE	0.8.1	ExpertCurated
_prototype_aromaticBOOL	Aromatic rings present (boolean)	ESCALATE	0.8.1	ExpertCurated
_prototype_AmineSitesINT	Number of ammonium sites	ESCALATE	0.8.1	ExpertCurated
_prototype_PrimarySitesBOOL	Primary amine present (boolean)	ESCALATE	0.8.1	ExpertCurated
_prototype_SecondarySitesBOOL	secondary amine present (boolean)	ESCALATE	0.8.1	ExpertCurated
_prototype_TertiarySitesBOOL	Tertiary amine present (boolean)	ESCALATE	0.8.1	ExpertCurated
_prototype_ImminiumSitesBOOL	Imminum site present (boolean)	ESCALATE	0.8.1	ExpertCurated
_prototype_heteroatomINT	number of heteroatom sites present	ESCALATE	0.8.1	ExpertCurated
_raw_inchikey	the inchikey of the organoammonium species in the reaction (for ESCALATE_report <0.8.1)	ESCALATE	0.8.1	ExpertCurated
_raw_smiles	the smiles string of a given species	ESCALATE	0.8.1	User Input

_feat_fr_amidine	Number of amidine groups	rdkit	Release_2019.09.1	(mol, countUnique=True, pattern=<rdkit.Chem.rdchem.Mol object at 0x7fb6ccb31d00>)
_feat_fr_Ar_NH	Number of aromatic amines	rdkit	Release_2019.09.1	(mol, countUnique=True, pattern=<rdkit.Chem.rdchem.Mol object at 0x7fb6ccb316e0>)
_feat_fr_ArN	Number of N functional groups attached to aromatics	rdkit	Release_2019.09.1	(mol, countUnique=True, pattern=<rdkit.Chem.rdchem.Mol object at 0x7fb6ccab9440>)
_feat_fr_dihydropyridine	Number of dihydropyridines	rdkit	Release_2019.09.1	(mol, countUnique=True, pattern=<rdkit.Chem.rdchem.Mol object at 0x7fb6ccab6e50>)

_feat_fr_guanido	Number of guanidine groups	rdkit	Release_2019.09.1	(mol, countUnique=True, pattern=<rdkit.Chem.rdchem.Mol object at 0x7fb6ccb31d70>)
_feat_fr_Imine	Number of Imines	rdkit	Release_2019.09.1	(mol, countUnique=True, pattern=<rdkit.Chem.rdchem.Mol object at 0x7fb6ccb317c0>)
_feat_fr_NH0	Number of Tertiary amines	rdkit	Release_2019.09.1	(mol, countUnique=True, pattern=<rdkit.Chem.rdchem.Mol object at 0x7fb6ccb31600>)
_feat_fr_NH1	Number of Secondary amines	rdkit	Release_2019.09.1	(mol, countUnique=True, pattern=<rdkit.Chem.rdchem.Mol object at 0x7fb6ccb31590>)

_feat_fr_NH2	Number of Primary amines	rdkit	Release_2019.09.1	(mol, countUnique=True, pattern=<rdkit.Chem.rdchem.Mol object at 0x7fb6ccb31520>)
_feat_fr_piperdine	Number of piperidine rings	rdkit	Release_2019.09.1	(mol, countUnique=True, pattern=<rdkit.Chem.rdchem.Mol object at 0x7fb6ccab6830>)
_feat_fr_piperzine	Number of piperazine rings	rdkit	Release_2019.09.1	(mol, countUnique=True, pattern=<rdkit.Chem.rdchem.Mol object at 0x7fb6ccab68a0>)
_feat_fr_pyridine	Number of pyridine rings	rdkit	Release_2019.09.1	(mol, countUnique=True, pattern=<rdkit.Chem.rdchem.Mol object at 0x7fb6ccab67c0>)
_feat_fr_quatN	Number of quaternary nitrogens	rdkit	Release_2019.09.1	(mol, countUnique=True, pattern=<rdkit.Chem.rdchem.Mol object at 0x7fb6ccb314b0>)