Supporting Information for

Efficiently Screening Organic Ligands by Machine Learning for Stabilizing 2D/3D Perovskites

Wei Zhang[a,b], Jing Zhang[c], Chengxu Zhang[a,b], Runchao Dong[b,d], Yong Xu[b], Zuobao Yang[b], Jinju Zheng[b], Dongsheng Chen[a], Qiao Liu[b]*, Weiyou Yangb and Ming-Hui Shang[b]**

[a] *College of Mathematics and Physics, Shanghai University of Electric Power, Shanghai 200090, P. R. China*

[b] *Institute of Micro/Nano Materials and Devices, Ningbo University of Technology, Ningbo, 315211, P. R. China*

[c] *School of Physical Science and Technology, Ningbo University, Ningbo, 315211, P. R. China*

[d] *School of Resources, Environment and Materials, State Key Laboratory of Featured Metal Materials and Life-cycle Safety for Composite Structures, Guangxi University, Nanning 530004, China*

* *Corresponding authors. E-mails: cds78@shiep.edu.cn (D. Chen), liuqiao@nbut.edu.cn (Q. Liu) and shangminghui@nbut.edu.cn (M.-H. Shang)*

This Supporting Information includes:

References

Supplementary text

Calculation of label y (E_b)

When constructing 2D capping layers, organic ligands need to embed into the perovskite surface to form a 2D coverage layer. To assess the stability of the 2D capping layer, we use the binding energy (E_b) of adjacent perovskite fragments. (Label y) The calculation formula is as follows:

$$E_b = E_{tot} - E_{fragment1} - E_{fragment2} \qquad \text{* MERGEFORMAT (S1)}$$

where E_{tot}, $E_{fragment1}$ and $E_{fragment2}$ are the total energies of the entire system, and two fragments cut from the optimized system.

Cesium vacancy formation energy ($E_{V_{Cs}}^{form}$)

Another method to evaluate the stability of the 2D capping layer is to verify whether it forms unwanted mixed phases under actual operating conditions.[1, 2] If the organic ligands in the 2D coverage layer further react, they will first replace the Cs atoms in the perovskite absorber layer, resulting in Cs vacancies. The difficulty of forming Cs vacancies can be measured by the vacancy formation energy. The calculation formula is as follows:

$$E_{V_{Cs}}^{form} = E_{Cs} + E_{slab} - E_{tot} \qquad \text{* MERGEFORMAT (S2)}$$

where E_{Cs} is the energy of a single Cs atom, E_{slab} is the energy of the defect model, and E_{tot} is the energy of the entire system.

Surface adsorption energy (E_{int})

After ensuring the stability of the 2D capping layer, we also need to verify the passivation effect of the organic ligands. Therefore, we calculated the interaction energy (Eint) between the organic ligands and the perovskite surface. The calculation formula is as follows:

$$E_{\text{int}} = E_{tot} - E_{slab} - E_{ligands} \qquad \text{* MERGEFORMAT (S3)}$$

where E_{tot} is the total energy of the system, E_{slab} is the energy of the surface model, and $E_{ligands}$ is the energy of a single organic ligand.

Pearson's correlation coefficient

Pearson's correlation coefficient is employed to eliminate redundant features. It evaluates the strength of the relationship between two features using the covariance matrix of the data.[3] The precise computational formula is as follows:

$$p = \frac{\sum_{i=1}^{n}\left[\left(x_i - \bar{x}\right)\times\left(y_i - \bar{y}\right)\right]}{\sqrt{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2}\times\sqrt{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}} \qquad \text{* MERGEFORMAT (S4)}$$

Where x_i and y_i are the values of two features in the i-th data point (i =1,2, 3…, n), and $\bar{x}$ and $\bar{y}$ are the average values of these two features across n data points. When the absolute value of p exceeds 0.8, it indicates a strong correlation between the two features. In such cases, we can select one representative feature and discard the other to reduce redundancy.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction and visualization tool that projects high-dimensional data into a lower-dimensional space through linear transformation while preserving the main features of the data.[4] Since our research field involves a small amount of data and

requires high data quality, it is best if the data used for training can represent a subset of the entire sample space. This can be clearly observed in a two-dimensional scatter plot with two principal components, PCA-1 and PCA-2. The principal component analysis was implemented using the scikit-learn package, and the detailed steps are as follows:

(a) Data standardization

(b) Calculation of the covariance matrix

(c) Eigenvalue decomposition

(d) Selection of principal components

Classification model

We used five common classification models: Linear Regression (LR), Random Forest (RF), Support Vector Classifier (SVC), Extreme Gradient Boosting Tree (XGBT), and Gaussian Naive Bayes (GNB).[5-7] These models were implemented based on the scikit-learn package. The specific meanings of each model are as follows:

(a) Linear Regression (LR) is a basic statistical method used to predict the linear relationship between a dependent variable and one or more independent variables. It fits the best line by minimizing the sum of squared errors, offering simplicity and strong interpretability. However, it is sensitive to outliers and can be affected by multicollinearity.

(b) Random Forest (RF) is an ensemble learning method that constructs multiple decision trees and combines their predictions for classification or regression. By randomly selecting features and data subsets to train each tree, it has strong resistance to overfitting and performs well with high-dimensional data.

(c) Support Vector Classifier (SVC) is a supervised learning model used for classification, which

separates different classes of data by finding a hyperplane that maximizes the margin between them. SVC performs well in high-dimensional spaces and is suitable for small sample sizes, but it is sensitive to the choice of parameters and kernel functions.

(d) Extreme Gradient Boosting Tree (XGBT) is an efficient tree model based on gradient boosting, which improves model accuracy by gradually adding weak learners. XGBT can handle missing values and nonlinear data, and it has performed exceptionally well in many machine learning competitions. However, it has a longer training time and complex parameter tuning.

(e) Gaussian Naive Bayes (GNB) is a classification algorithm based on Bayes' theorem, which assumes that features are independent of each other. Although this assumption often does not hold in reality, GNB performs well with high-dimensional data and small sample sizes, offering fast computation and simple implementation.

K-fold cross-validation

K-fold cross-validation is a commonly used model evaluation method that divides the dataset into K equal parts (called "folds") for multiple rounds of training and testing, ensuring more stable and reliable evaluation results for the model.[8] The basic idea is to repeatedly split the dataset and validate on different training and testing sets, thereby reducing the bias caused by data partitioning. The steps to implement K-fold cross-validation are as follows:

(a) Data Partitioning: Randomly divide the entire dataset into K equal parts, each called a "fold".

(b) Iterative Training and Testing:

a) For each fold i (from 1 to K):

i. Use the i-th fold as the test set and remaining K-1 folds as the training set.

ii. Train the model on the training set.

iii. Evaluate the model performance on the test set and record the evaluation results (e.g., accuracy, mean squared error, etc.)

(c) Result Aggregation: Calculate the average or other statistics of the K evaluation results as the final performance metric of the model.

The advantages of K-fold cross-validation include making full use of every sample in the dataset, leading to more stable and reliable model evaluation, especially suitable for situations with small data volumes. Additionally, by multiple rounds of training and testing, it can effectively identify overfitting or underfitting issues in the model, thereby better tuning the model parameters.

Grid search and hyperparameter optimization

Grid search is a systematic hyperparameter optimization method that performs an exhaustive search over a predefined parameter grid to find the combination of parameters that yields the best model performance. Specifically, grid search traverses all possible parameter combinations and evaluates each combination's performance through cross-validation, thereby selecting the optimal parameter set. Hyperparameter optimization is crucial for improving model performance because appropriate hyperparameters can significantly enhance the model's generalization ability, reduce the risk of overfitting, and improve prediction accuracy. By combining grid search with ten-fold cross-validation, we can comprehensively evaluate the model under different parameter combinations, ensuring that the final selected model performs optimally on both the training and test sets. This process not only improves the model's robustness but also enhances its ability to handle different feature descriptors (discrete and continuous features), thereby making the model more reliable and stable in practical applications.

Our dataset has the following unique characteristics: the dataset is relatively small (as shown in Table S1), and the input variables include both discrete and continuous feature descriptors (for example, descriptors like atom counts are discrete, while molecular weight is continuous). To evaluate the model's accuracy, the dataset was randomly divided into a training set and a test set in a 4:1 ratio. To avoid overfitting, we used grid search and ten-fold cross-validation to select the model's hyperparameters and plotted the learning curves for each model.

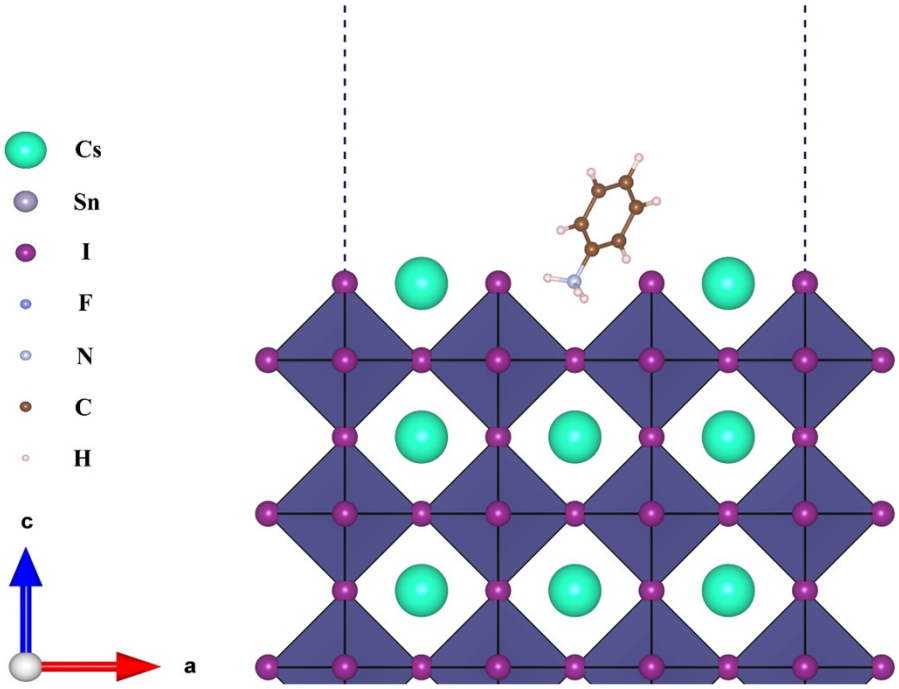
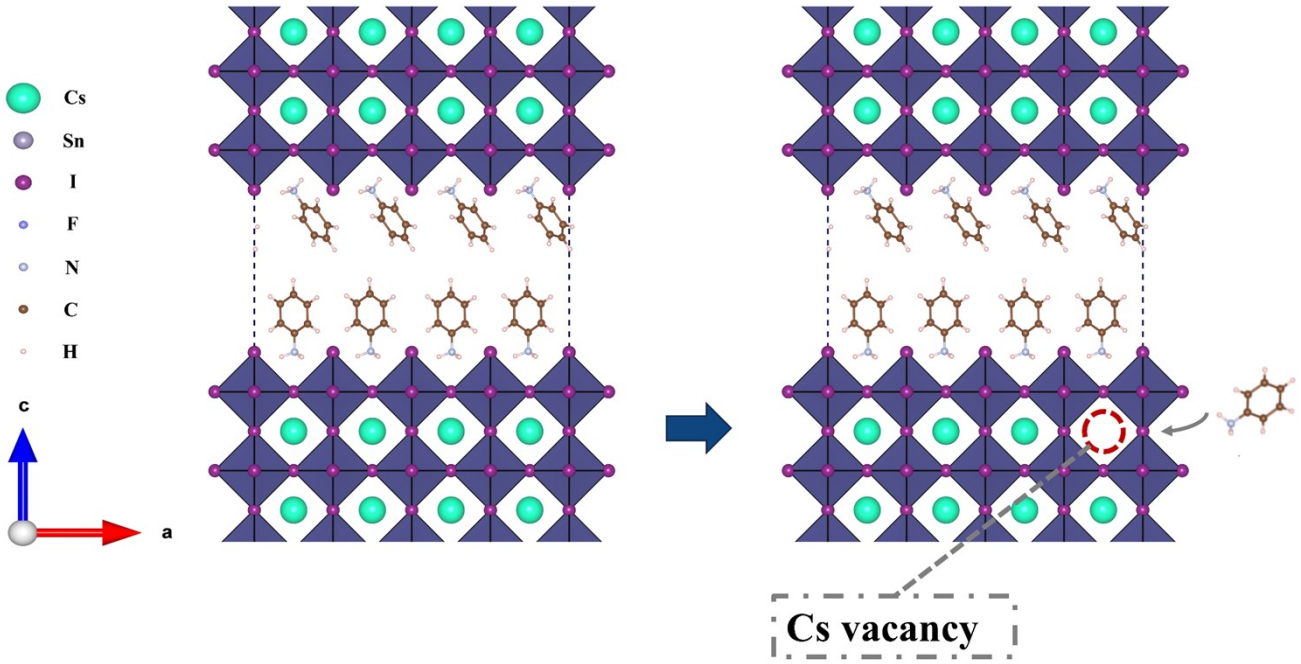


Figure S1. The interaction energy model between organic ligand and perovskite surface was calculated.



135

136 **Figure S2.** A model for calculating Cs vacancy formation energy.

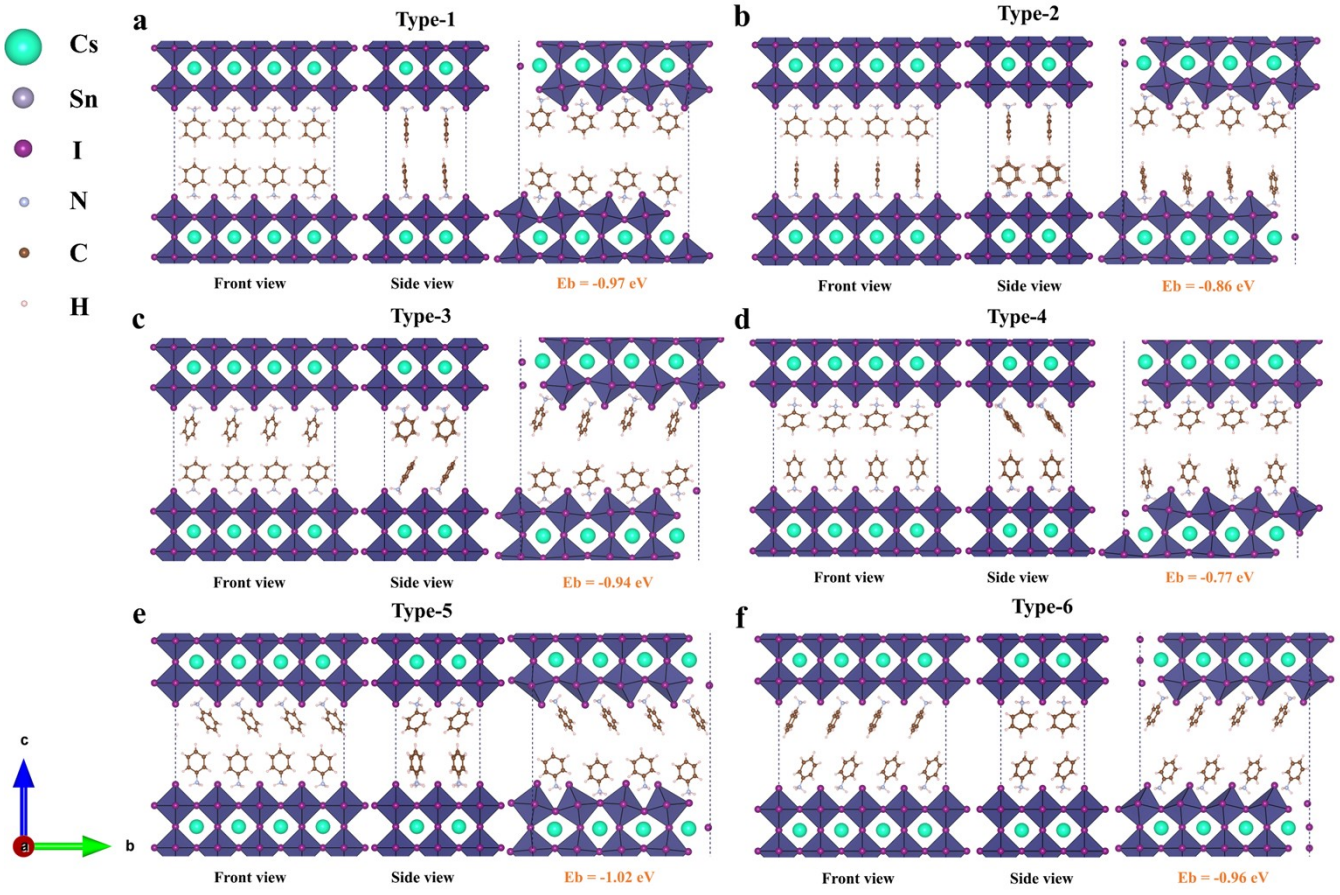


Figure S3. Possible configurations of organic ligands between adjacent perovskite fragments: (a) Type-1. (b) Type-2. (c) Type-3. (d) Type-4. (e) Type-5. (f) Type-6.

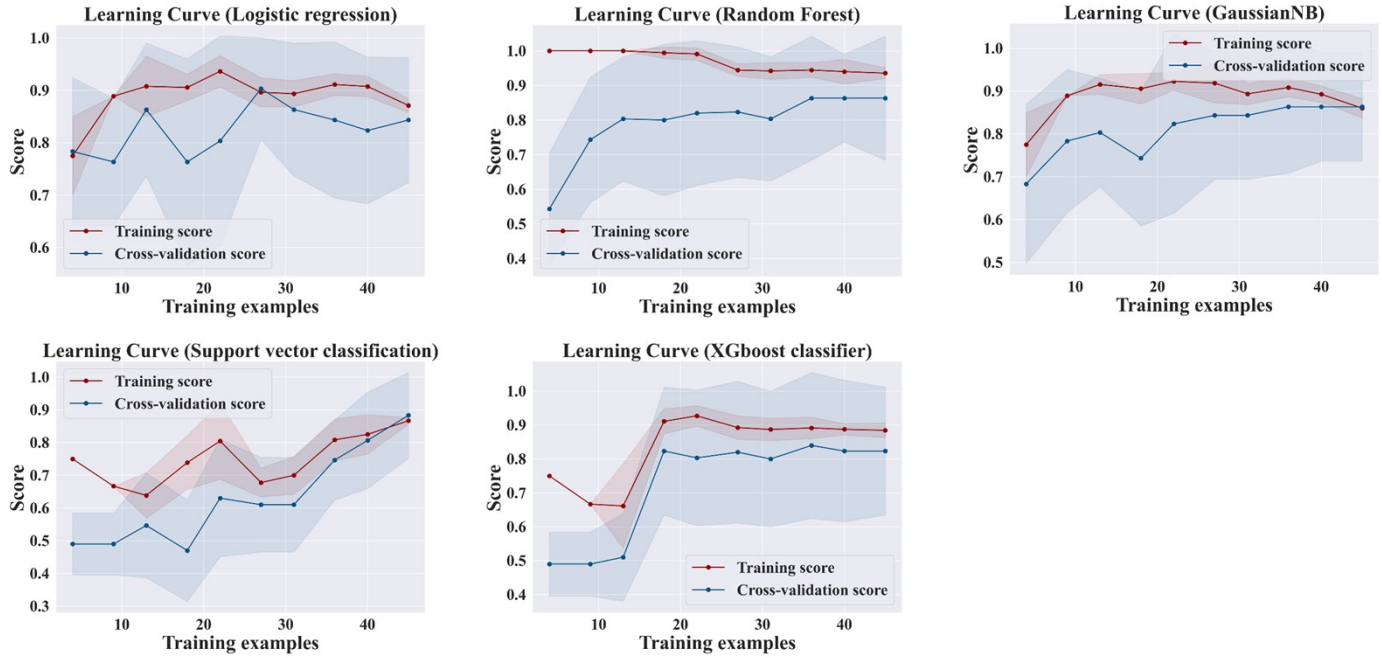


Figure S4. Learning curves for the five models: Logistic Regression, Random Forest, Support Vector Machine, XGBoost, and Naive Bayes.

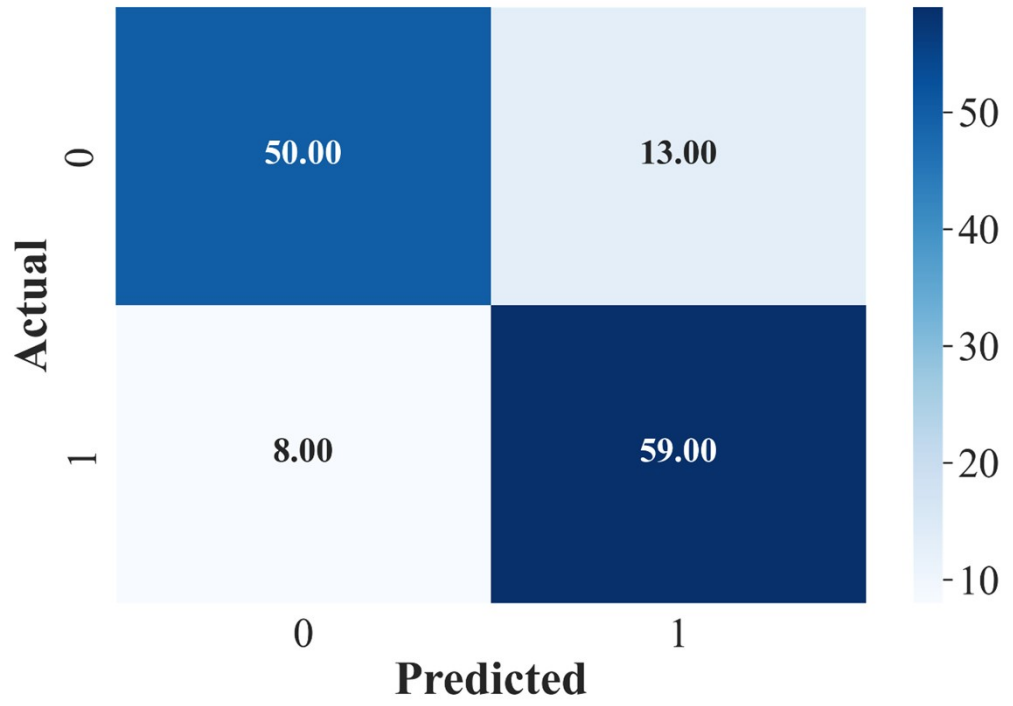


Figure S5. Performance of the Random Forest model on ten different test sets.

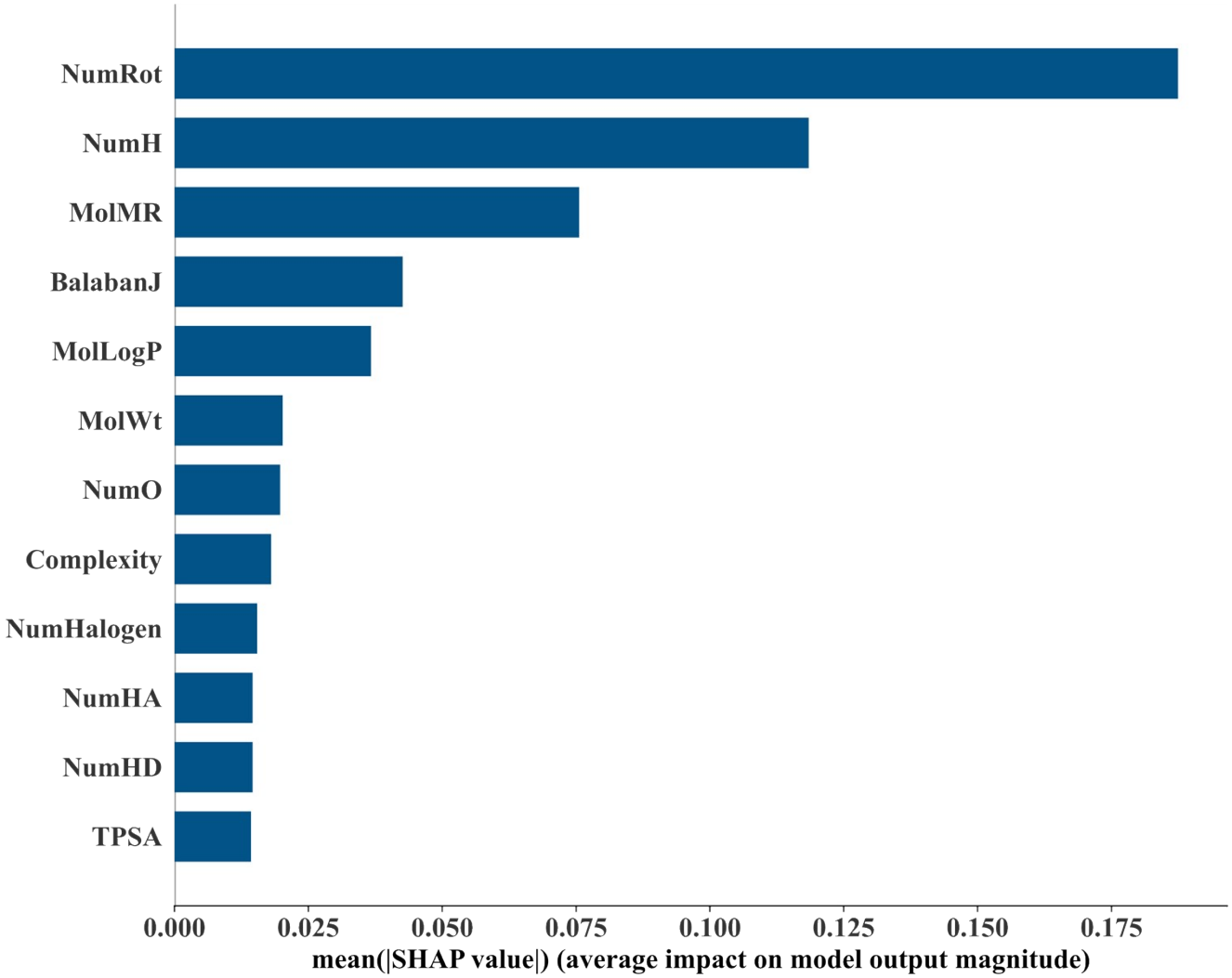


Figure S6. The random forest model outputs a feature importance ranking of class 1.

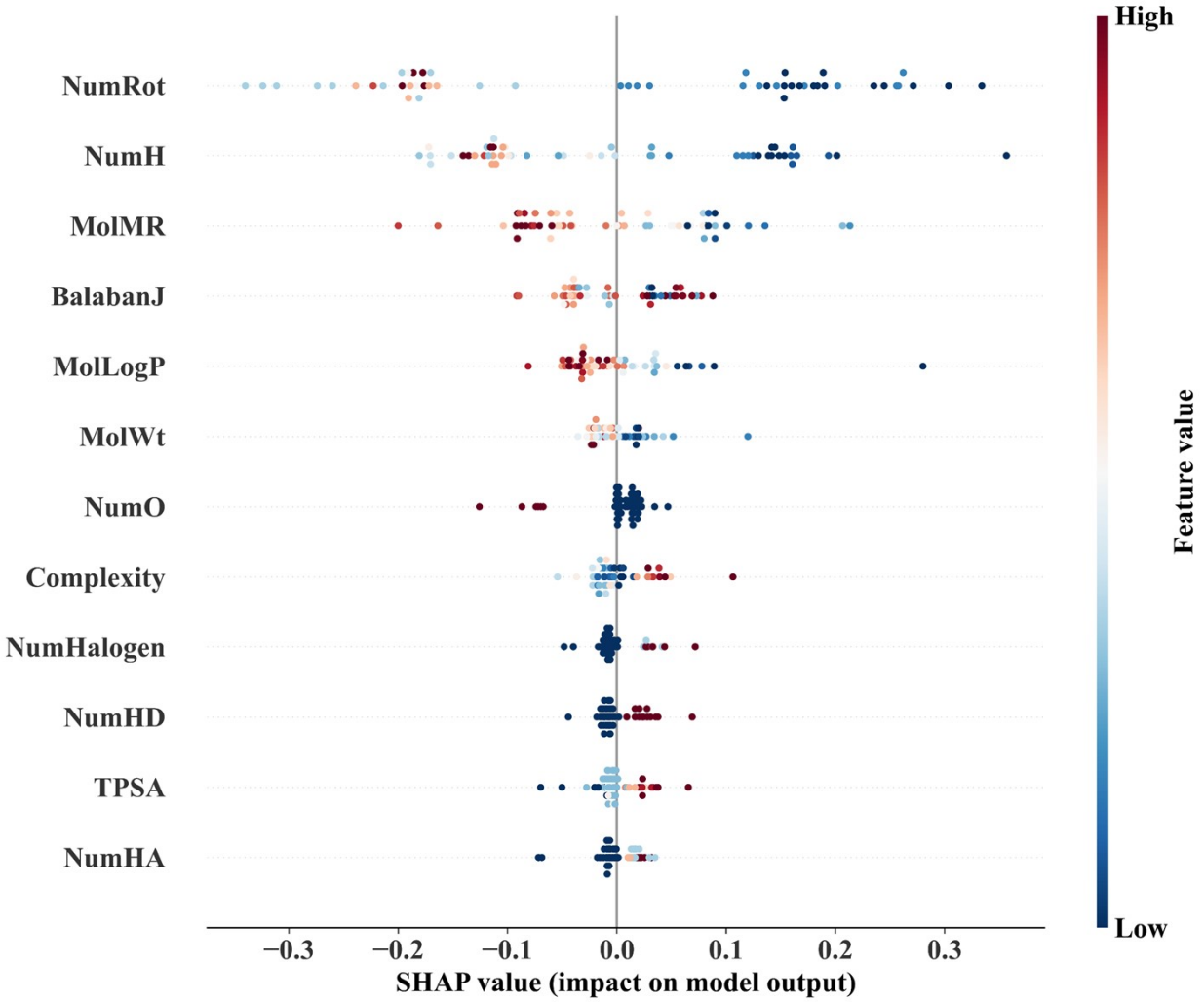


Figure S7. The contribution values of all features.

158
159

160 **Tabel S1.** Results of label *y* calculation for 64 samples

ID	Name	E_{tot}	$E_{fragment1}$	$E_{fragment2}$	E_b	Label *y*
0	2-phenylethylazanium	-677.19	-184.21	-491.77	-1.22	0
1	butylazanium	-533.95	-184.06	-348.66	-1.23	0
2	octylazanium	-799.35	-183.82	-613.89	-1.64	0
3	Phenylazanium(An^+)	-542.96	-183.98	-357.95	-1.03	1
4	(4-fluorophenyl)azanium	-544.03	-183.88	-359.02	-1.02	1
5	(3,4-difluorophenyl)azanium	-544.12	-183.81	-359.15	-0.96	1
6	(3,4,5-trifluorophenyl)azanium	-543.91	-183.90	-359.12	-0.90	1
7	3-azaniumylpropylazanium	-352.41	-184.42	-167.75	-0.24	1
8	propylazanium	-467.63	-184.13	-282.32	-1.18	0
9	2-phenylpropylazanium	-743.32	-184.33	-557.95	-1.04	0
10	2-(4-methylphenyl)ethylazanium	-743.98	-184.25	-558.51	-1.22	0
11	1-aminoethylideneazanium	-421.02	-183.67	-236.47	-0.88	1
12	cyclohexylmethylazanium	-703.42	-184.21	-518.06	-1.14	0
13	tert-butylazanium	-534.28	-183.49	-349.56	-1.22	0
14	benzylazanium	-610.36	-184.19	-425.33	-0.84	1
15	hexan-2-ylazanium	-666.81	-183.39	-481.40	-2.02	0
16	4-methylpentan-2-ylazanium	-667.11	-183.62	-481.63	-1.87	0
17	2-(4-fluorophenyl)ethylazanium	-678.23	-184.19	-492.81	-1.23	0
18	2-[4-	-748.74	-184.30	-563.33	-1.10	0

	(trifluoromethyl)phenyl]ethylazaniu m					
19	[4-(trifluoromethyl)phenyl]methylazan ium	-681.83	-184.11	-496.97	-0.74	1
20	[4-(trifluoromethyl)phenyl]azanium	-614.67	-183.90	-429.79	-0.98	1
21	[amino-(4-fluorophenyl)methylidene]azanium	-631.09	-183.77	-448.36	1.03	1
22	ethylazanium	-401.18	-184.10	-216.08	-1.00	1
23	hexylazanium	-666.57	-184.04	-481.28	-1.25	0
24	diaminomethylideneazanium	-404.23	-183.87	-219.54	-0.82	1
25	[amino(phenyl)methylidene]azaniu m	-630.05	-183.81	-447.47	1.23	1
26	(3-fluorophenyl)methylazanium	-611.45	-184.15	-426.34	-0.96	1
27	azetidin-1-ium	-432.55	-184.41	-247.72	-0.42	1
28	2-(4-methoxyphenyl)ethylazanium	-768.66	-184.25	-583.21	-1.19	0
29	2-methylpropylazanium	-534.03	-184.58	-348.72	-0.73	1
30	4-carboxybutylazanium	-625.58	-184.01	-440.15	-1.41	0
31	2-(2,3,4,5,6-pentafluorophenyl)ethylazanium	-678.58	-184.02	-493.64	-0.92	1
32	8-azaniumyloctylazanium	-519.14	-184.15	-333.77	-1.21	0
33	2-azaniumylethylazanium	-318.95	-184.03	-134.57	-0.35	1

34	di(propan-2-yl)azanium	-665.43	-181.21	-481.25	-2.97	0
35	4-ethylmorpholin-4-ium	-658.16	-182.98	-473.77	-1.41	0
36	octan-3-ylazanium	-798.93	-183.92	-613.51	-1.50	0
37	cyclohexylazanium	-636.54	-184.04	-451.82	-0.68	1
38	dipropylazanium	-663.98	-180.10	-480.96	-2.92	0
39	prop-2-enylazanium	-433.48	-184.34	-248.58	-0.56	1
40	3-methoxypropylazanium	-557.76	-184.13	-372.51	-1.12	0
41	3-propan-2-yloxypropylazanium	-691.44	-184.09	-506.16	-1.19	0
42	3-(2-methoxyethoxy)propylazanium	-714.71	-183.96	-529.44	-1.31	0
43	(4-azaniumylphenyl)azanium	-390.60	-184.11	-205.75	-0.74	1
44	tripropylazanium	-862.40	-181.19	-678.49	-2.72	0
45	(4-methoxyphenyl)methylazanium	-702.01	-184.20	-516.73	-1.09	0
46	dimethylazanium	-399.78	-183.74	-215.01	-1.02	1
47	12-azaniumyldodecylazanium	-651.77	-183.91	-466.30	-1.55	0
48	1-phenylethylazanium	-676.92	-183.88	-491.46	-1.57	0
49	(4-ethenylphenyl)methylazanium	-710.47	-184.21	-525.69	-1.06	1
50	heptan-2-ylazanium	-733.16	-183.49	-547.67	-2.00	0
51	but-3-enylazanium	-500.61	-184.18	-315.03	-1.41	0
52	1-(4-chlorophenyl)ethylazanium	-671.56	-183.74	-486.12	-1.70	0
53	(4-chlorophenyl)-methylazanium	-603.32	-183.52	-418.33	-1.46	0
54	2-ethylhexylazanium	-799.54	-184.41	-613.91	-1.22	0

55	pyridin-1-ium-3-ylmethanamine	-590.28	-183.98	-410.38	4.09	1
56	pyridin-1-ium-4-ylmethanamine	-590.74	-183.94	-409.83	3.03	1
57	2-bromoethylazanium	-393.02	-184.41	-207.64	-0.97	1
58	2-iodoethylazanium	-390.58	-183.90	-206.07	-0.60	1
59	2-[3-(trifluoromethyl)phenyl]ethylazanium	-747.80	-183.72	-563.64	-0.43	1
60	1H-benzimidazol-3-ium	-599.08	-183.84	-415.46	0.22	1
61	piperidin-1-ium-4-ylmethanamine	-683.97	-183.90	-498.35	-1.72	0
62	pyrrolidin-1-ium-3-amine	-550.26	-183.84	-365.07	-0.95	0
63	amino(methyl)azanium	-377.77	-184.08	-193.12	-0.56	1

Table S2. The hyperparameters of LR, RF, SVC, XGBT and GNB.

Classification model	Hyperparameters
LR	C = 0.01, penalty = 'l2', solver = 'liblinear'
RF	max_depth = 2, min_samples_split = 2, n_estimators = 42, random_state = 42
SVC	C = 0.01, degree = 2, gamma = 0.05, kernel = 'linear'
XGBT	colsample_bytree=1.0, gamma=0, learning_rate=0.01, max_depth=3, n_estimators=47, subsample=0.6
GNB	None

Table S3. Highly efficient organic ligands reported experimentally between 2022 and 2024.

Name	Molecular formula	CID	Num Rot	Num H	MolM R	Balaban J	References	Model predict
4-BBA$^+$	$C_6H_7BrN^+$	23277843	0	7	36.7	3.03	Ref. [9] Energy Environ. Sci., **2022**,15, 5168-5180	1
PAH$^+$	$C_4H_7N_4^+$	6343289	1	7	28.7	2.93	Ref. [10] Adv. Funct. Mater., **2023**, 33, 37, 2303038	1
DFPD$^+$	$C_5H_{10}F_2N^+$	7006488	0	10	25.8	2.33	Ref. [11] ACS Energy Lett. **2024**, 9, 2, 363-372	0
TFPhFA$^+$	$C_8H_8F_3N_2^+$	6339126	1	8	41.7	3	Ref. [12] Adv. Funct. Mater., **2022,** 33, 4, 2209921	0
MOR$^+$	$C_4H_{10}NO^+$	5020115	0	10	22.5	2	Ref. [13] J. Am. Chem. Soc. **2024**, 146, 11, 7555-7564	1
SMO$^+$	$C_4H_{10}NS^+$	6997275	0	10	29.0	2	Ref. [13] J. Am. Chem. Soc. **2024**, 146, 11, 7555-7564	1

Tabel S4. Feature descriptor values and prediction results of 83 samples

ID	NumH	NumHalogen	NumO	MolWt	NumHD	NumHA	NumRot	TPSA	Complexity	BalabanJ	MolLogP	MolMR	proba_0	proba_1	labely
0	28	0	0	186.36	1	0	10	27.6	81.2	2.76	3.15	59.60	0.86	0.14	0
1	16	0	0	246.3	1	0	3	4.4	202	2.12	3.87	79.27	0.78	0.22	0
2	16	0	0	198.28	1	0	2	16.6	154	2.15	2.83	63.60	0.60	0.40	0
3	16	0	2	230.28	1	2	4	35.1	184	2.08	2.23	67.23	0.86	0.14	0
4	20	0	0	130.25	1	0	6	16.6	37.8	2.60	1.15	41.48	0.80	0.20	0
5	20	0	0	178.29	0	0	4	0	124	3.01	3.05	59.86	0.75	0.25	0
6	14	0	1	188.25	1	1	3	36.9	172	2.43	1.46	57.20	0.82	0.18	0

7	14	0	0	172.25	1	0	1	27.6	167	2.76	2.14	55.25	0.56	0.44	0
8	16	0	0	198.28	1	0	3	27.6	148	2.34	2.06	62.48	0.81	0.19	0
9	20	0	0	190.3	1	0	3	27.6	155	2.12	2.20	58.90	0.80	0.20	0
10	18	0	0	212.31	1	0	4	27.6	178	2.13	2.25	67.08	0.79	0.21	0
11	24	0	0	218.36	1	0	4	27.6	178	1.77	2.81	68.01	0.77	0.23	0
12	7	0	0	83.11	1	0	0	19.7	44.8	3.05	-0.16	21.97	0.18	0.82	1
13	11	0	2	167.19	1	2	2	73.5	148	2.78	0.38	44.56	0.45	0.55	1
14	18	0	2	172.24	2	2	3	53.9	151	2.25	0.21	45.85	0.68	0.32	0
15	8	0	2	90.1	2	2	2	64.9	52.8	2.83	-1.30	20.01	0.24	0.76	1
16	10	0	2	104.1	2	2	2	64.9	72.1	3.34	-1.05	24.55	0.28	0.72	1

				3										
17	11	1	0	140.18	1	1	2	27.6	95.3	2.75	0.61	37.86	0.41	0.59	1
18	16	0	0	102.2	1	0	3	4.4	25.7	2.99	-0.07	32.59	0.68	0.32	0
19	14	0	0	136.21	1	0	3	27.6	74.8	2.43	0.86	42.52	0.80	0.20	0
20	14	0	0	88.17	1	0	3	27.6	19.9	2.34	0.42	27.28	0.63	0.37	0
21	5	0	0	45.064	2	0	0	51.6	7.5	2.19	-2.27	12.18	0.11	0.89	1
22	12	0	0	74.14	1	0	1	27.6	19.6	2.54	0.03	22.64	0.37	0.63	1
23	9	0	0	109.15	2	1	0	53.7	72.9	3.13	0.14	33.40	0.19	0.81	1
24	16	0	0	150.24	1	0	4	27.6	84.9	2.28	1.25	47.14	0.80	0.20	0
25	11	1	0	140.18	1	1	2	27.6	95.3	2.81	0.61	37.86	0.41	0.59	1

26	9	1	0	126.15	1	1	1	27.6	77	2.88	0.57	32.99	0.26	0.74	1
27	5	0	0	69.09	2	0	0	29.9	24.1	3.13	-0.17	17.35	0.13	0.87	1
28	5	0	0	69.09	1	1	0	29	73	2.61	-0.94	19.56	0.17	0.83	1
29	6	0	0	80.11	1	0	0	14.1	30.9	3.00	0.50	23.00	0.26	0.74	1
30	8	0	1	62.09	2	1	1	47.9	10	1.97	-1.78	14.84	0.15	0.85	1
31	8	0	0	114.19	1	1	1	55.9	56	2.76	0.49	30.91	0.24	0.76	1
32	21	0	0	145.27	2	1	7	53.7	55.2	2.65	0.53	44.51	0.68	0.32	0
33	10	0	2	152.17	2	2	2	64.9	141	2.85	0.05	39.71	0.36	0.64	1
34	9	0	0	61.11	2	1	1	53.7	8	1.97	-1.81	16.81	0.05	0.95	1
35	10	0	0	60.12	1	0	0	27.6	10.8	2.32	-0.36	18.02	0.25	0.75	1
36	12	0	0	86.16	1	0	0	16.6	30.9	2.00	-0.27	25.51	0.35	0.65	1
37	20	0	0	130.2	1	0	3	4.4	59	3.68	0.71	41.78	0.67	0.33	0

				5											
38	8	0	1	122.14	0	1	1	21	103	3.15	0.32	33.01	0.29	0.71	1
39	5	1	0	159	1	0	0	14.1	56	3.02	1.26	30.70	0.31	0.69	1
40	36	0	0	242.46	1	0	14	27.6	123	2.85	4.71	78.07	0.86	0.14	0
41	12	0	0	170.23	1	0	2	16.6	116	2.16	2.21	54.13	0.64	0.36	0
42	14	0	1	116.18	1	1	1	25.8	59.5	2.13	-0.64	31.69	0.47	0.53	1
43	12	0	2	106.14	1	2	3	46.1	36.7	2.99	-1.15	25.56	0.59	0.41	0
44	5	5	0	150.07	1	5	1	27.6	94.9	4.14	0.43	18.76	0.21	0.79	1
45	6	6	0	182.09	1	6	2	16.6	98.7	3.44	0.67	23.77	0.26	0.74	1

46	16	0	2	134.2	1	2	6	35.1	44.3	2.60	-1.16	35.41	0.75	0.25	0
47	12	0	2	106.14	3	2	4	57.1	28.9	2.45	-2.47	25.83	0.54	0.46	0
48	6	1	0	129.57	2	1	0	40.2	76.8	3.03	0.74	32.42	0.19	0.81	1
49	12	0	0	74.14	1	0	2	16.6	11.1	2.19	-0.41	23.01	0.41	0.59	1
50	28	0	0	186.36	1	0	9	4.4	72.1	3.47	2.27	60.29	0.75	0.25	0
51	28	0	0	186.36	0	0	8	0	80.2	4.19	3.44	60.89	0.75	0.25	0
52	36	0	0	242.46	0	0	12	0	116	4.39	5.00	79.36	0.75	0.25	0
53	6	0	0	32.065	1	0	0	27.6	2	1.00	-1.14	8.81	0.10	0.90	1
54	22	0	0	144.28	1	0	7	27.6	52.7	2.65	1.98	45.75	0.87	0.13	0

55	24	0	0	158.3	1	0	8	27.6	61.9	2.69	2.37	50.37	0.87	0.13	0
56	18	0	0	118.22	2	0	5	55.3	31.5	2.53	-0.97	33.98	0.58	0.42	0
57	13	1	0	214.07	1	0	4	27.6	31.3	2.45	0.83	40.49	0.74	0.26	0
58	14	0	1	152.21	1	1	3	36.9	106	2.74	0.48	44.45	0.79	0.21	0
59	15	1	0	228.09	1	0	5	27.6	39.5	2.53	1.22	45.11	0.77	0.23	0
60	12	0	0	158.22	1	0	1	27.6	144	2.67	1.58	50.54	0.53	0.47	0
61	8	0	0	58.1	1	0	0	27.6	22.5	2.17	-0.61	15.91	0.17	0.83	1
62	14	0	0	172.25	1	0	2	27.6	155	2.49	1.62	55.41	0.58	0.42	0
63	8	0	0	70.11	1	0	1	27.6	47.9	2.48	-0.75	21.21	0.19	0.81	1
64	10	0	0	92.19	1	1	2	52.9	16.4	2.19	-0.41	26.14	0.24	0.76	1

65	14	0	0	208.28	1	0	1	27.6	216	2.56	2.73	68.05	0.56	0.44	0
66	16	0	0	198.28	1	0	3	27.6	164	2.24	2.14	63.34	0.79	0.21	0
67	20	0	0	178.29	1	0	6	27.6	106	2.07	2.03	56.37	0.85	0.15	0
68	12	0	0	242.4	1	3	4	109	177	2.10	2.81	66.11	0.70	0.30	0
69	16	0	1	202.27	1	1	4	36.9	183	2.26	1.85	61.82	0.86	0.14	0
70	9	0	0	109.15	1	1	1	40.5	63.5	2.83	-0.18	30.83	0.21	0.79	1
71	9	0	0	109.15	1	1	1	40.5	63.5	2.83	-0.18	30.83	0.21	0.79	1
72	18	0	0	212.31	1	0	4	27.6	176	2.05	2.06	67.15	0.79	0.21	0
73	22	0	0	204.3	1	0	3	27.6	166	2.09	2.59	63.51	0.80	0.20	0

				3											
74	11	3	0	190.19	1	3	2	27.6	155	3.05	1.49	42.90	0.36	0.64	1
75	10	2	0	158.17	1	2	2	27.6	109	2.89	0.75	37.82	0.34	0.66	1
76	6	2	0	82.07	1	2	1	27.6	21.6	2.54	-0.51	13.76	0.16	0.84	1
77	5	3	0	100.06	1	3	0	27.6	38.5	3.17	-0.21	13.81	0.15	0.85	1
78	15	0	0	115.2	2	1	1	42.6	63.5	2.13	-1.08	33.44	0.31	0.69	1
79	11	0	0	87.14	2	1	0	28.6	32.5	2.00	-1.85	24.53	0.20	0.80	1
80	4	1	0	170.961	2	0	0	51.6	27.5	2.80	-1.50	25.14	0.18	0.82	1
81	12	0	2	118.15	2	2	3	64.9	82.5	3.52	-0.66	29.17	0.56	0.44	0
82	10	0	2	104.13	2	2	3	64.9	62.7	2.82	-0.91	24.62	0.42	0.58	1

Table S5. Photovoltaic parameters of 2D perovskite solar cell.

Organic ligands	V_{oc} (V)	J_{sc} (mA/cm^2)	*FF* (%)	PCE (%)
Control An $^+$	0.74	12.49	45.75	4.23
222tA $^+$	0.78	19.28	60.89	9.16
2amA $^+$	0.66	23.94	63.84	10.09

Code S1. Feature descriptor extraction code.

```
import pandas as pd

from rdkit import Chem

from rdkit.Chem import Descriptors



# Read the CSV file containing SMILES strings

df = pd.read_csv('your_dataset', encoding='utf-8', index_col="ID")

smiles_list = df["SMILES"].tolist()



# Create an empty list to store molecular descriptors

descriptors_list = []



# Define the required molecular descriptors and their corresponding functions

selected_descriptors = {

    "MolWt": Descriptors.MolWt,

    "NumHD": Descriptors.NumHDonors,

    "NumHA": Descriptors.NumHAcceptors,

    "NumRot": Descriptors.NumRotatableBonds,

    "TPSA": Descriptors.TPSA,

    "BalabanJ": Descriptors.BalabanJ,

    "MolMR": Descriptors.MolMR,

    "MolLogP": Descriptors.MolLogP,
```

```
    "Ipc": Descriptors.Ipc
}

# Calculate molecular descriptors for each molecule and add to the list
for smiles in smiles_list:
    mol = Chem.MolFromSmiles(smiles)
    descriptors = [3]
    for desc_name, desc_func in selected_descriptors.items():
        try:
            value = desc_func(mol)
            descriptors[desc_name] = value
        except Exception:
            descriptors[desc_name] = None
    descriptors_list.append(descriptors)

# Convert the list of molecular descriptors to a DataFrame
descriptors_df = pd.DataFrame(descriptors_list)

# Save the molecular descriptors to a CSV file
descriptors_df.to_csv("feature.csv", index=False)
```

Code S2. Model training part of the code

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score, learning_curve

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from xgboost import XGBClassifier

from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import confusion_matrix, accuracy_score

import seaborn as sns

import matplotlib.pyplot as plt



df = pd.read_csv('your_dataset.csv',index_col='ID')



X = df.drop('labely', axis=1)

y = df['labely']



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)



scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)



models = [

    ('Logistic Regression', LogisticRegression()),

    ('Random Forest', RandomForestClassifier(random_state = 42)),

    ('Support Vector Classifier', SVC()),

    ('XGBoost', XGBClassifier()),

    ('Gaussian Naive Bayes', GaussianNB())

]



for model_name, model in models:

    cv_scores = cross_val_score(model, X_train, y_train, cv=10)

    print(f'{model_name} Cross-validation scores: {cv_scores}')

    print(f'{model_name} Mean cross-validation score: {cv_scores.mean()}')



print('_____

_____')



classifiers = {

    'Logistic Regression': (LogisticRegression(), {

        'penalty': ['l2'],
```

```
        'C': [0.01, 0.1, 1, 10, 100],
        'solver': ['liblinear'],
        'class_weight': [None, 'balanced']
    }),
    'Random Forest': (RandomForestClassifier(random_state=42), {
        'n_estimators': range(36, 45),
        'max_depth': range(1, 5),
        'min_samples_split': range(2, 10),
    }),
    'Support Vector Classifier': (SVC(), {
        'C': [0.01, 0.1, 1, 10, 100],
        'kernel': ['linear', 'rbf'],
        'gamma': [0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14],
        'degree': [2, 3, 4],
        'class_weight': [None, 'balanced']
    }),
    'XGBoost': (XGBClassifier(), {
        'n_estimators': range(40, 50),
        'max_depth': [3, 6, 9],
        'learning_rate': [0.01, 0.1, 0.3],
        'subsample': [0.6, 0.8, 1.0],
        'colsample_bytree': [0.6, 0.8, 1.0],
```

```
            'gamma': [0, 0.1, 0.2]
        }),
        'GaussianNB': (GaussianNB(), {
        })
}


def train_and_evaluate(classifier, param_grid, X_train, y_train, X_test, y_test):
        grid_search = GridSearchCV(classifier, param_grid, cv=10, scoring='accuracy', n_jobs=-1, error_score='raise')
        try:
                grid_search.fit(X_train, y_train)
                print(f'Best parameters found: {grid_search.best_params_}')
                print(f'Best cross-validation score: {grid_search.best_score_}')


                best_model = grid_search.best_estimator_
                best_model.fit(X_train, y_train)


                y_train_pred = best_model.predict(X_train)
                y_test_pred = best_model.predict(X_test)


                train_accuracy = accuracy_score(y_train, y_train_pred)
                test_accuracy = accuracy_score(y_test, y_test_pred)
```

```
            print(f'Training accuracy: {train_accuracy}')
            print(f'Testing accuracy: {test_accuracy}')
        except Exception as e:
            print(f"An error occurred: {e}")

for name, (classifier, param_grid) in classifiers.items():
    print(f'\n{name}')
    train_and_evaluate(classifier, param_grid, X_train, y_train, X_test, y_test)

print('_____
_____')
```

References

1 A. A. Sutanto, N. Drigo, V. I. E. Queloz, I. Garcia-Benito, A. R. Kirmani, L. J. Richter, P. A. Schouwink, K. T. Cho, S. Paek, M. K. Nazeeruddin and G. Grancini, *J. Mater. Chem. A*, 2020, **8**, 2343-2348.

2 J. Chakkamalayath, N. Hiott and P. V. Kamat, *ACS Energy Lett.*, 2022, **8**, 169-171.

3 J. Ma, S. Zhang, X. Liu and J. Wang, *Bioresour. Technol.*, 2023, **389**, 129820.

4 C. J. Bartel, *Patterns N. Y.*, 2021, **2**, 100382-100384.

5 W. Sun, Y. Zheng, K. Yang, Q. Zhang, A. A. Shah, Z. Wu, Y. Sun, L. Feng, D. Chen, Z. Xiao, S. Lu, Y. Li and K. Sun, *Sci. Adv.*, 2019, **5**, 4275-4282.

6 C. C. Chang and C. J. Lin, *Acm. T. Intel. Syst. Tec.*, 2011, **2**, 1-27.

7 T. K. Ho, J. Hull and S. N. Srihar, *IEEE Trans. Pattern Anal. Mach. Intell.*, 1994, **14**, 66-75.

8 B. P. Burman, *Biometrika*, 1989, **76**, 503-514.

9 Y. Yan, R. T. Wang, Q. S. Dong, Y. F. Yin, L. H. Zhang, Z. H. Su, C. Y. Wang, J. S. Feng, M. H. Wang, J. Liu, H. R. Ma, Y. L. Feng, W. Z. Shang, Z. Y. Wang, M. Z. Pei, Y. D. Wang, S. Y. Jin, J. M. Bian, X. Y. Gao, S. Z. Liu and Y. T. Shi, *Energ. Environ. Sci.*, 2022, **15**, 5168-5180.

10 L. Liu, J. Tang, S. Li, Z. Yu, J. Du, L. Bai, X. Li, M. Yuan and T. Jiu, *Adv. Funct. Mater.*, 2023, **33**, 2303038.

11 B. H. Chang, L. Wang, H. Li, L. Pan, Y. T. Wu, Z. Liu, Y. N. Zhang, E. Y. Guo and L. W. Yin, *ACS Energy Lett.*, 2024, **9**, 363-372.

12 X. Yue, X. Zhao, B. Fan, Y. Yang, L. Yan, S. Qu, H. Huang, Q. Zhang, H. Yan, P. Cui, J. Ji, J. Ma and M. Li, *Adv. Funct. Mater.*, 2022, **33**, 2209921.

13 T. Wang, L. Bi, L. Yang, Z. Zeng, X. Ji, Z. Hu, S. W. Tsang, H. L. Yip, Q. Fu, A. K. Jen and Y. Liu, *J. Am. Chem. Soc.*, 2024, **146**, 7555-7564.