ChBE 6745 Project: final report

Prediction of Adsorption Properties of Metal-Organic Frameworks with Framework Flexibility

Chao-Wen Chang, Pengfei Cheng, Po-Wei Huang, Xiaohan Yu, Yamin Zhang

April 18, 2020

1. Introduction

1.1. Project description

Metal-organic frameworks (MOFs) are organic-inorganic hybrid crystalline porous materials, consisting of a regular array of positively charged metal ions surrounded by organic ligand molecules. MOFs offer unique structural diversity: varying metal ions and organic ligands can change its porosity, geometry, topology and chemical functionality. This makes it possible to find a MOF to selectively adsorb specific gases, but also makes it a hard task to search through the diverse collection of MOFs.

The diversity of MOFs motivates the application of molecular simulation for efficiently, systematically predicting the adsorption properties of certain molecules. However, most simulations neglect the structural flexibility of MOFs for computational efficiency, which includes thermal vibrations and linker rotation. Agrawal and Sholl (2019) show that appropriate consideration of framework flexibility may significantly affect the results of adsorption uptakes and selectivities. However, taking the framework flexibility into account directly in the molecular simulation framework may be 5-10 times more computationally expensive than standard simulation methods based on rigid crystal structure. Therefore, we are interested in constructing fast, accurate regression model to predict the adsorption properties of MOFs with framework flexibility involved.

1.2. Goals

We plan to construct different regression models and evaluate their performance on *predicting single-component adsorption uptake of MOFs with framework flexibility with respect to certain adosrbates at saturation pressure*, based on three types of input:

1. the features of MOFs,
2. the features of adsorbates, and
3. the single-component adsorption uptake of MOFs with framework flexibility with respect to certain adosrbates from standard simulation methods based on rigid crystal structure.

The main focus of this project is to:

1. implement various regression algorithms on the proposed dataset to find the one with the best prediction ability,

2. apply hyperparameter tuning for each algorithm to improve model performance, and
3. combine different validation strategies and apply throughout the project to make the models robust and least biased.

The features of the MOF have been pre-selected and the number of the overall features is relatively small, so we do not address feature engineering as the main focus of our project except for the LASSO models.

2. Methodology

2.1. Dataset details

The dataset discussed below is provided by Dr. David Sholl's group. The corresponding files can be found at the `data` folder.

The dataset is composed of three parts:

1. 28 MOF features (under `data/ML_data`),
2. 6 adsorbate features (manually added in the `.ipynb` file), and
3. adsorption uptakes of 801 (MOF, adsorbate) pairs (under `data/flexibility_data/y_data/adsorption_data`), containing two values:
   a) values from rigid model (serve as a feature)
   b) mean values from flexible model (serve as y)

A visualization of these uptake values is given in Figure 1.

All features are continuous entries. The dimension of the input is $(801, 28 + 6 + 1) = (801, 35)$. The dimension of the output is $(801, 1)$.

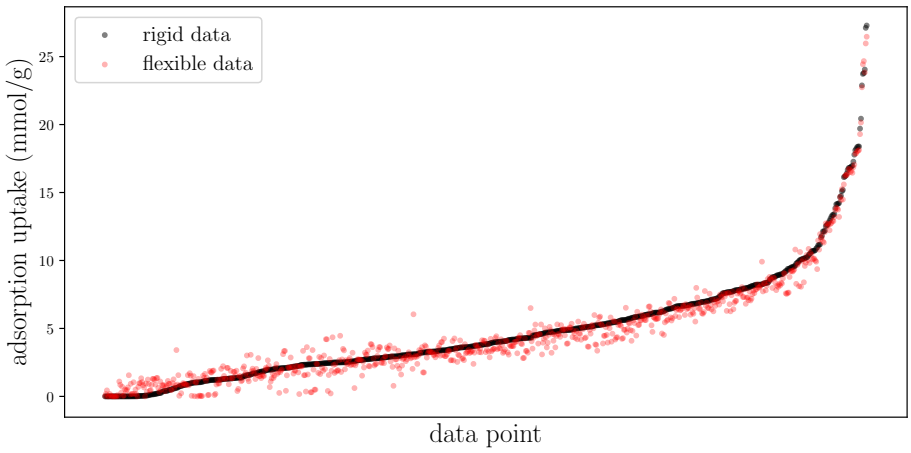

Figure 1: Visualization of uptake values from rigid model and model considering framework flexibility

2.2. Model training strategy

Five regression models have been chosen for the project:

1. Multi-linear regression
2. RBF-kernel regression (without regularization)
3. Kernel ridge regression (KRR)
4. LASSO regression
5. MLPRegressor (neural network)

The multi-linear model serves as a baseline model to assess the performance of other models. The RBF kernel regression model (without regularization) also serves as a baseline model with nonlinearity.

The rest three models are implemented as improved models. The KRR model is built to show the power of regularization compared with the pure RBF kernel regression model. The LASSO regularization is combined with both multi-linear regression and RBF kernel regression, which can be used to see the impact of complexity optimization on the prediction performance. Finally the MLPRegressor model is built as a representation of the neural network (NN) algorithms, and implementing this model is an attempt on trying to learn the mechanism of NN models, which hyperparameters they have, and how to tune them.

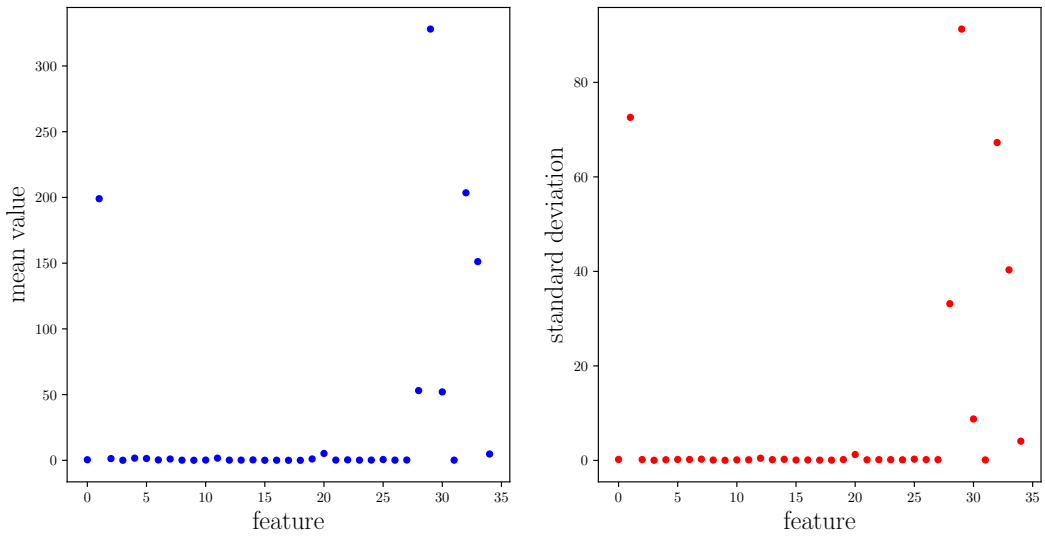

Figure 2: Summary statistics of features

The mean values and standard deviations of the features are shown in Figure 2. It can be seen clearly that the scales of these features are different, which means that it is necessary to scale the data before training regression models. Standard scaling is based on the training-testing set of the data, and is applied to both the training-testing set and the validation set of the data to avoid potential contamination. The two data sets are further described in the next section. Then both the unscaled data and the scaled data are used for each model, except MLPRegressor, to see if the data scaling will have significant impact on model performance. MLPRegressor only uses the scaled data because it may cause some numerical error in the training process when the magnitudes of different features differ too much.

2.3. Model validation strategy

The validation strategy is mainly composed of two parts:

1. We first apply hold-out to split 25% of the data as validation set, which would not be modified or changed throughout the training-testing process.
2. After hold-out, we then apply 5-fold on the rest 75% data to form five (training set, test set) pairs. Each training set contains $75\% \times 80\% = 60\%$ data, and each test set contains $75\% \times 20\% = 15\%$ data. The same five pairs are used throughout the project for model training and hyperparameter tuning for all models. In this way the training and the tuning processes will be least biased by how the dataset is split into the training and test sets.

The 5-fold is implemented based on the following feature of `GridSearchCV`: it can take CV splitter object as its `cv` argument. Therefore, if we can define a 5-fold CV splitter object beforehand and apply it during the training process for all the models, we can guarantee that the same (training set, test set) pairs are used and no bias is generated by the validation strategy. For models that do not use `GridSearchCV`, an explicit for-loop iterating the five (training set, test set) pairs is used instead, which are generated from the same CV splitter object.

2.4. Hyperparameter tuning strategy

A general hyperparameter optimization strategy is used for all models except for `MLPRegressor`. This strategy is composed of two steps:

1. rough estimation: for each hyperparameter, a list of values across a wide range (logarithmically discretized) is given to `GridSearchCV`. `GridSearchCV` iterates through the pairs of all values and returns the best ones using the 5-fold cross validation described above.
2. refined estimation: a second round of `GridSearchCV` is implemented, with the range of hyperparameter values limited to the around best hyperparameters found in the rough estimation step.

This strategy assumes that the loss function is convex over all hyperparameters jointly, in which a second round of `GridSearchCV` can find a model no worse than the one found in the first round. This is a strong assumption and are not true for most cases, but addressing it will be well beyond the scope of this course.

For MLPRegressor, there are three main types of hyperparameters:

1. the number of hidden layers ($L \in \mathbb{N}$),
2. the number of neurons in each hidden layer ($n_l \in \mathbb{N}, l \in \{1, 2, ..., L\}$), and
3. activation function types (Logistic function or Rectified Linear Unit (ReLU) function).

The forms of two activation functions are:

$$\mathcal{L}_{\text{logistic}}(\boldsymbol{\varepsilon}) = \sum_i \max(0, \varepsilon_i) \tag{1}$$

$$\mathcal{L}_{\text{ReLU}}(\boldsymbol{\varepsilon}) = \sum_i \ln(1 + e^{-\varepsilon_i}) \tag{2}$$

The number of parameters in the model grows linearly with n_l and exponentially with L. Therefore, it

may be computationally expensive to simultaneously tune n_l and L. An alternative way is to first tune n_1 for a single-layer NN model, then fix n_1 and tune n_2 and so on. So we propose to tune the hyperparameter for the NN model using the alternative way while limiting the upper bound of n_i to 100 and L to 3. The tuning process will be conducted twice with Logistic and ReLU as activation functions respectively.

2.5. Block diagram of data pipeline and model

The block diagram of the project is shown in Figure 3. First we pre-process the data from three datasets described in Section 2.1 to generate a uniform dataset (X, y). Then we subsequently apply the hold-out and 5-fold described in Section 2.3, after which we obtain five pairs of (training set, test set) and use them to train five models mentioned in Section 2.2. Finally the five models are compared together using the validation data that are held out in the beginning.

2.6. Software packages

All alogrithms and functions used in the project are imported from `sklearn`.

- multi-linear regression: `sklearn.linear_model.LinearRegression`
- RBF-kernel regression: `sklearn.metrics.pairwise.rbf_kernel`
- KRR: `sklearn.kernel_ridge.KernelRidge`
- LASSO: `sklearn.linear_model.Lasso`
- MLPregressor: `sklearn.neural_network.MLPRegressor`
- validation strategy: `sklearn.model_selection.train_test_split`, `sklearn.model_selection.KFold`
- hyperparameter tuning: `sklearn.model_selection.GridSearchCV`

3. Results

The computational results of implementing five regression models listed with the validation data are presented here. There are four factors that are considered for each model:

- training cost,
- r^2,
- mean absolute error (MAE), and
- Bayesian information criteria (BIC).

Among these four factors, the training cost reflects the computational cost of the models; r^2 and MAE both assess the prediction performance of the models, with the former as a unitless, relative measure and the latter as an absolute measure with units; BIC accounts for the degree of complexity for the models.

The implementation and explanation of model training, data preprocessing and other related parts are given in the `.ipynb` file with details.

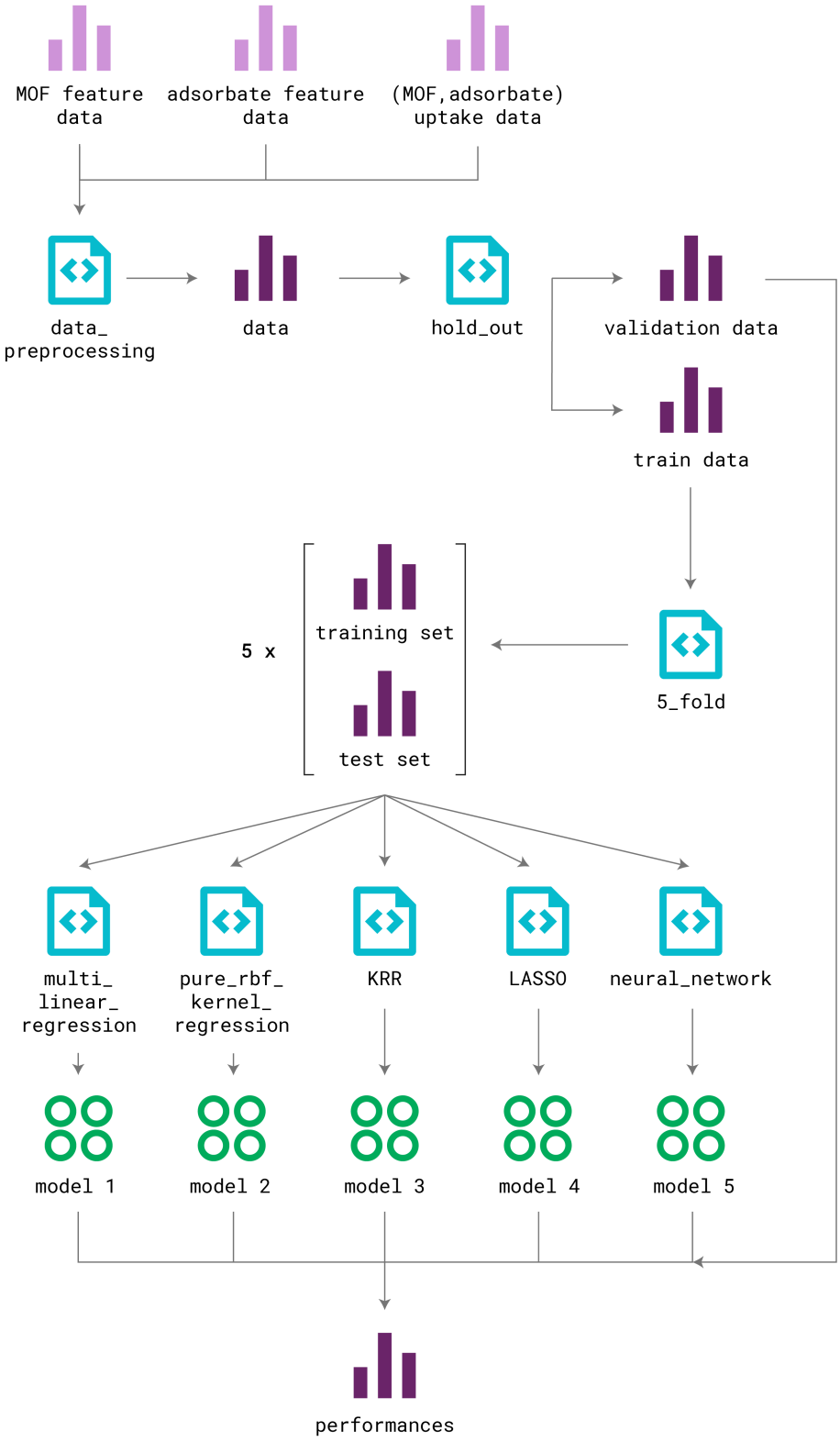

Figure 3: Project block diagram

3.1. Training cost

The time consumed when models are trained is recorded below as the computational cost of training for each model. Since the hyperparameter tuning strategies vary for different models, both the total training time and the average training time are given in Table 1 and Table 2.

The number of iterations in the table corresponds to the total fitting times of the model during a single call in the training process. As the number of hyperparameters differs in each model, the number of iterations also significantly differs among all the models. All models are run on an Intel Core i9 (8 cores) 3.10 GHz processor with 32 GB RAM. Python 3.7 and `sklearn` 0.20.1.

For each model in Table 1, the training cost using the unscaled data and the scaled data are almost identical. Among different models, multi-linear regression takes the least amount of time, as it is the simplest model. LASSO-multi-linear regression takes the second least amount of time for the same reason. RBF, LASSO-RBF-kernel and KRR models take significantly longer time, as the data is lifted into the kernel space, which has higher dimensions than the original feature space. Among them LASSO-RBF and KRR take less time, indicating that the regularization term in the loss function helps shorten the optimization time (though it does not directly indicate that it makes the optimization simpler, as the solver may converge to a local optimum that performs poorly).

Table 1: Training cost of each model (except MLPRegressor)

	data set	total time (s)	iterations	average time (s)
multi-linear regression	unscaled	0.0054	5	0.0011
	scaled	0.0055	5	0.0011
RBF kernel regression	unscaled	2.15	101	0.0213
	scaled	2.40	101	0.0238
KRR	unscaled	23.80	1325	0.0180
	scaled	19.20	1325	0.0145
LASSO-multi-linear	unscaled	0.19	95	0.0020
	scaled	0.18	95	0.0019
LASSO-RBF-kernel	unscaled	5.00	1220	0.0041
	scaled	5.02	1220	0.0041

All MLPRegressor models take training time one order of magnitude larger than the kernel-based models. The reason is as mentioned in Section 2.4 that the number of parameters in an MLPRegressor model grows linearly with the number of neurons and exponentially with the number of hidden layers, which significantly increases the model size. For example, for a 2-layer MLPRegressor model with 50 neurons for the first layer and 60 neurons for the second layer, it may require around 3000 parameters for the single model. For the models with the same activation function, the training cost increases with the increase of the number of layers. For model with different activation functions, models with ReLU typically spend less time than models with logistic. Again, the convergence characteristics of the MLPRegressor models do not necessarily correspond to their prediction performance, as they may be caused by the nature of their function forms.

A bar chart of training time of all models are shown in Figure 4. The models can be categorized into three

Table 2: Training cost of MLPRegressor

number of layers	activation function	total time (s)	iterations	average time (s)
1	logistic	17.8	100	0.178
	ReLU	18.1	100	0.181
2	logistic	32.3	100	0.323
	ReLU	19.8	100	0.198
3	logistic	57.9	100	0.579
	ReLU	26	100	0.26

groups with different scales of training time: MLPRegressor models, kernel-based models and multi-linear models. The performance of these models is investigated in the next section.

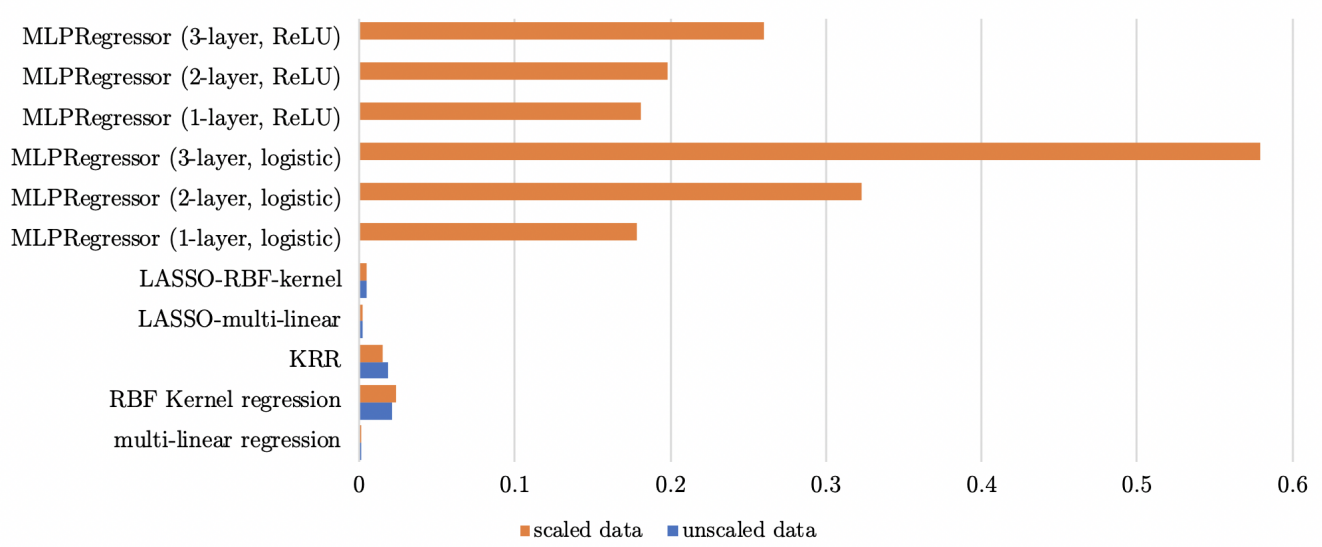

Figure 4: Average training time (s)

3.2. r^2

After the training process, all models are tested using the validation data to calculate r^2 values. The results are given in Table 3. For models trained by the unscaled data, two multi-linear models and the KRR model outperforms the RBF kernel regression model and the LASSO-RBF-kernel model. This may indicate that there exists some nice linear correlations among features and the output such that even the basic multi-linear regression model manages to obtain a very high r^2 value. The poor performance of the RBF kernel regression model may be caused by the overfitting of the model, which can be seen when it is compared with the KRR model. When the regularization term is added to the loss function, which can avoid the overfitting by penalizing the ℓ_2 norm of the parameters, the KRR model has a significantly

higher r^2 value. The LASSO-RBF-kernel obtains the worst r^2 value. The reason may be caused by the local optimum issue mentioned above. The r^2 values for the training-testing process for each model are also listed in the table for reference. These values are close to the validation r^2 values except for the LASSO-RBF-kernel model with unscaled data.

Table 3: r^2 of all models using validation data

	validation r^2		training-testing r^2	
	unscaled data	scaled data	unscaled data	scaled data
multi-linear regression	0.978	0.978	0.989	0.989
RBF Kernel regression	0.767	0.894	0.829	0.898
KRR	0.965	0.993	0.963	0.987
LASSO-multi-linear	0.979	0.977	0.978	0.978
LASSO-RBF-kernel	0.288	0.925	0.445	0.933
MLPRegressor (1-layer, logistic)		0.976		0.959
MLPRegressor (2-layer, logistic)		0.975		0.961
MLPRegressor (3-layer, logistic)		0.975		0.961
MLPRegressor (1-layer, ReLU)		0.963		0.960
MLPRegressor (2-layer, ReLU)		0.962		0.961
MLPRegressor (3-layer, ReLU)		0.955		0.964

For the models using the scaled data, they all obtain high r^2 values (> 0.9). Compared with the unscaled-data models, it indicates that feature scaling is important at least for the RBF kernel regression model and the LASSO-RBF-kernel model. For two multi-linear models, feature scaling does not have big impact on the performance. The KRR model manages to obtain the highest r^2 with the scaled data, while it is already very good with the unscaled data.

For all the MLPRegressor models, their performance is similar to each other no matter how many layers they have or which activation function they use. This indicates that for the dataset of this project the 1-layer MLPRegressor is already good enough, and adding number of hidden layer may not improve the model performance correspondingly with more time devoted into the model. Compared with other methods, the MLPRegressor models shows average prediction ability while spending much more training time (Figure 4).

A corresponding bar chart is shown in Figure 5, where the y-axis is limited to $[0.9, 1]$. In the plot the RBF kernel regression model and the LASSO-RBF-kernel model with unscaled data are not shown due to their performance caused by the reason analyzed above. Among other models, they all perform quite well, and there is no single model that is significantly better than others, although the KRR model with scaled data does outperform others with r^2 even close to 1.

3.3. MAE

The MAE values are also calculated for all models using the validation data as the second accuracy criterion with the unit of the uptake remained. The results are shown in Table 4 and Figure 6, which share some trends with the r^2 results: feature scaling significantly improved the performance of kernel-based models; The results of MLPRegressor are similar for varying activation functions and number of hidden

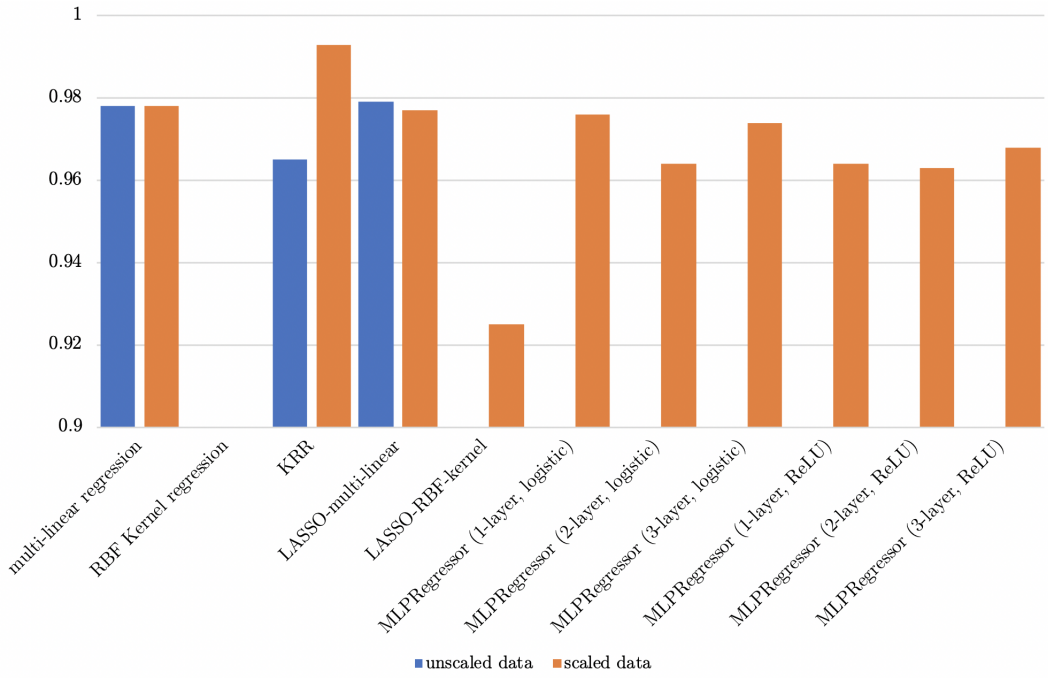

Figure 5: r^2 of all models using validation data

layers. A main difference shown in Figure 6 is that the KRR model with scaled data has a significantly smaller value of MEA compared with other models. This indicates that the KRR model with scaled data is able to predict the uptake with less absolute errors.

Table 4: MAE of all models (mmol/g)

	unscaled data	scaled data
multi-linear regression	0.424	0.424
RBF Kernel regression	1.374	0.701
KRR	0.532	0.231
LASSO-multi-linear	0.413	0.436
LASSO-RBF-kernel	2.320	0.775
MLPRegressor (1-layer, logistic)		0.494
MLPRegressor (2-layer, logistic)		0.451
MLPRegressor (3-layer, logistic)		0.451
MLPRegressor (1-layer, ReLU)		0.569
MLPRegressor (2-layer, ReLU)		0.617
MLPRegressor (3-layer, ReLU)		0.606

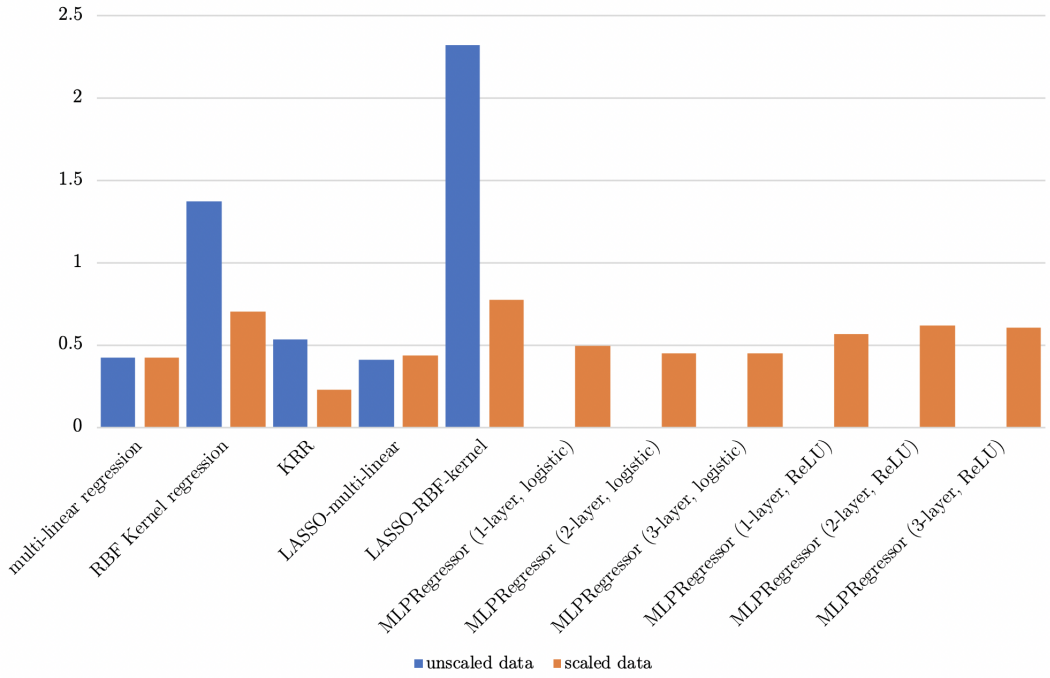

Figure 6: MAE of all models using validation data (mmol/g)

3.4. Bayesian information criterion

Although BIC is not used as a criterion during the training process, it may still be a good indicator of the complexity of the final models, especially when the prediction performance of models is similar. It is calculated using the following equation:

$$\mathrm{BIC} = n \cdot \ln\left({\sigma_e}^2\right) + k \cdot \ln\left(n\right) \tag{3}$$

where n is the number of data points, k is the number of parameters and σ_e is the standard deviation of errors.

The BICs of all models are given in Table 5. whether data is scaled or not does not have a significant effect on BIC, except for the RBF kernel regression model, which has a much worse score with scaled data. For the LASSO-RBF-kernel model its BICs are similar for both datasets as its BICs are dominated by the number of parameters and the impact of errors is reduced. Among different models, it follows the same rough categories mentioned in Section 3.1: multi-linear models have the lowest BICs, the MLPRegressor models have the highest BICs, while the kernel-based models rank in the middle. For kernel-based and MLPRegressor models, the values of BICs are mainly determined by the number of parameters, and models among each group basically have the same values of BICs.

Table 5: BIC of all models

	unscaled data	scaled data
multi-linear regression	-62	-62
RBF Kernel regression	4.11E+02	2.62E+03
KRR	3.04E+03	2.72E+03
LASSO-multi-linear	8.52E+02	8.53E+02
LASSO-RBF-kernel	3.76E+03	3.85E+03
MLPRegressor (1-layer, logistic)		1.90E+04
MLPRegressor (2-layer, logistic)		6.38E+04
MLPRegressor (3-layer, logistic)		1.07E+05
MLPRegressor (1-layer, ReLU)		1.10E+04
MLPRegressor (2-layer, ReLU)		1.42E+04
MLPRegressor (3-layer, ReLU)		1.80E+04

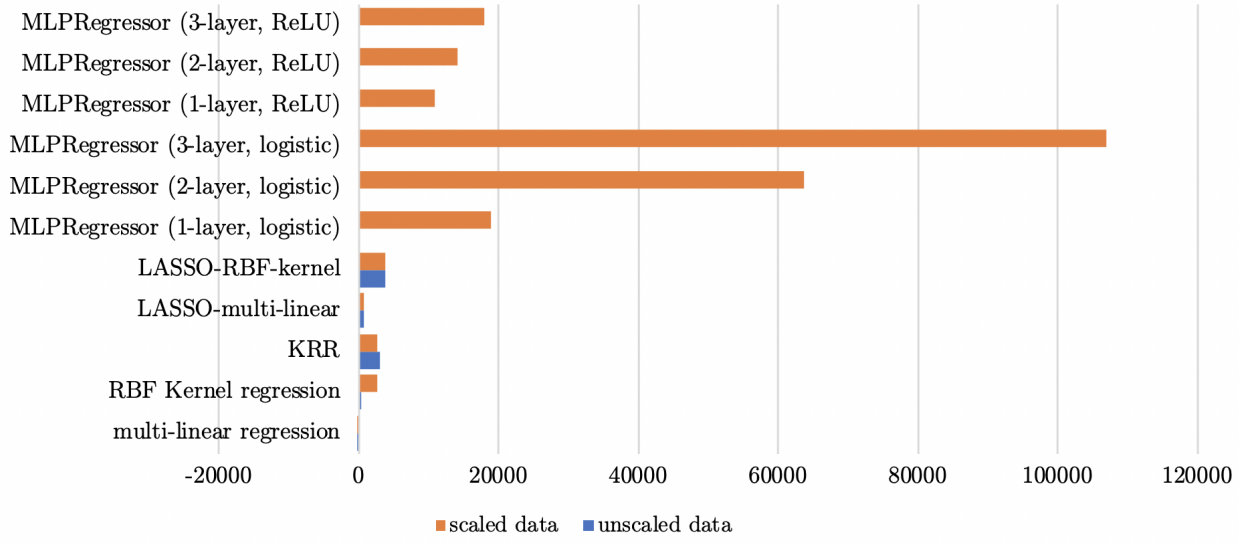

Figure 7: BIC of all models

4. Discussion

4.1. KRR: size and distribution of parameters

To avoid overfitting, smoother models are preferred. We look at how using different input data (unscaled and scaled) affects the size and distribution of the parameters in KRR models. Histograms are drawn and the largest absolute values of the parameters in KRR using unscaled and scaled data are printed below.

The parameters are normally distributed for two models. The values of parameters in KRR models, which include regularization terms in the loss function, are in a reasonable range. One thing to notice is that the size of the parameters is one order of magnitude smaller after scaling the data by comparing the largest

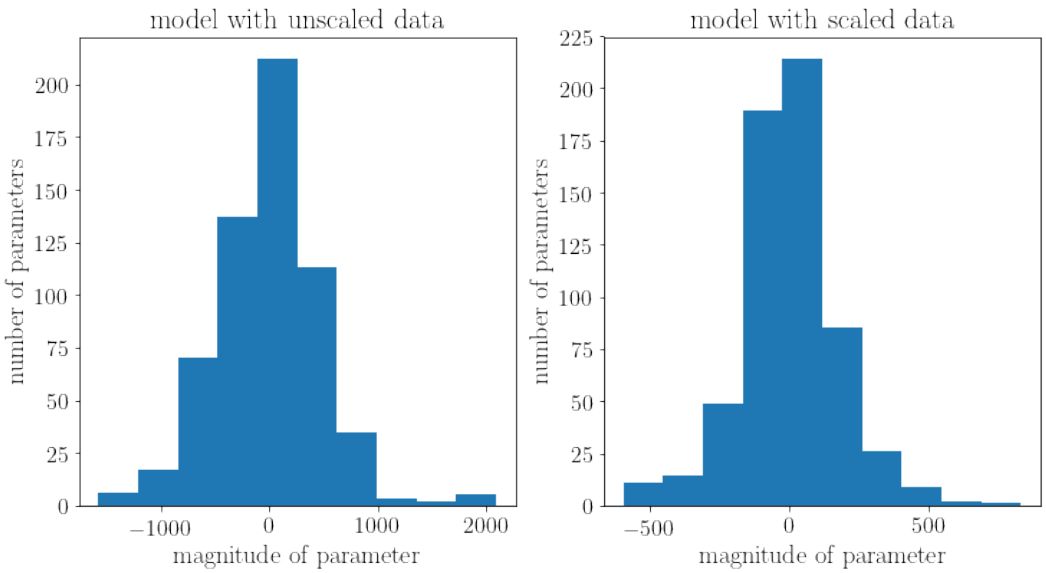

Figure 8: Histograms of parameter magnitudes of KRR models

parameter in the model. This indicates that KRR using scaled data is more likely to be not overfitted with high accuracy.

4.2. Comparison of kernel-based models

The forms of loss functions of three kernel-based methods are as follows:

$$\mathcal{L}_{\mathrm{RBF}} = \sum_i {\varepsilon_i}^2 \tag{4}$$

$$\mathcal{L}_{\mathrm{KRR}} = \sum_i {\varepsilon_i}^2 + \alpha \left\|\mathbf{w}\right\|_2 \tag{5}$$

$$\mathcal{L}_{\mathrm{LASSO}} = \sum_i {\varepsilon_i}^2 + \alpha \left\|\mathbf{w}\right\|_1 \tag{6}$$

To avoid overfitting, different regularization terms are added in KRR ($\alpha \left\|\mathbf{w}\right\|_2$) and LASSO ($\alpha \left\|\mathbf{w}\right\|_1$) loss functions. It is interesting and worthwhile to compare the results of these three methods.

First the hyperparameters in three models are compared. RBF can be written in the form of Gaussian distribution as a function of γ:

$$\mathrm{rbf}(i,j) = \exp(-\gamma(x_i - x_j)^2) = \exp\left(\frac{-(x_i-\mu)^2}{2\sigma^2}\right) \tag{7}$$

where $\gamma = \frac{1}{2\sigma^2}$, which is inversely proportional to the width of Gaussian distribution σ. The larger γ is, the narrower a feature, then the features are less correlated. The γ from three models using scaled and unscaled data are as follows:

γ	unscaled data	scaled data
RBF	8E-08	2E-02
KRR	7E-05	5E-03
LASSO	3E-03	2E-02

Table 6: γ of three kernel-based models

For models using unscaled data, γ in RBF is orders of magnitude smaller than γ in KRR and LASSO, which means that features in this method are least correlated. After scaling the data, the gamma values for all three methods are larger, which means the features are more correlated and corresponds to the correlation matrix drawn above.

In KRR and LASSO-RBF-kernel, another hyperparameter is included, which is α. α is an indication of regularization strength. When α is very small, there's nearly no regularization, and when α is extremely large, the size of coefficients is important while the error square summation is ignored.

α	unscaled data	scaled data
KRR	8E-04	3E-03
LASSO	8E-04	5E-05

Table 7: α of kernel-based models

α values for KRR and LASSO using both scaled and unscaled data are close.

Then we look at the size of the parameters in the models(largest absolute value of the parameters):

	unscaled data	scaled data
RBF	9E+14	5E+04
KRR	3E+03	4E+02
LASSO	2E+01	8E+00

Table 8: largest parameter of kernel-based models

For RBF, we have extremely large parameters, in which case the model tends to be overfitting. After adding the regularization terms to the loss function, the sizes of the parameters for KRR and LASSO are significantly smaller. Though we have relatively small αs, regularization is playing an important role in the regression while maintaining a high accuracy in KRR and LASSO. The size of the parameters in LASSO is orders of magnitude smaller than that of KRR. Using scaled the data, the largest parameters in all three models are smaller. In conclusion, the analysis on hyperparameters emphasizes the importance of regularization and scaling.

4.3. LASSO-multi-linear regression: dropping features

When we explored the LASSO-multi-linear regression model with different configurations, we noticed that the model may drop most of the features while keeping relatively good accuracy. For example, when the tolerance of `Lasso` instance is set to 0.1, the LASSO-multi-linear model with scaled data only remains 6 features with $r^2 = 0.968$. The remaining features are:

- the uptake value from simulations considering rigid structures,
- MOF-pore volume per unit cell,
- MOF-total number of atoms in the cell,
- MOF-number of 6-member rings,
- adsorbate-critical pressure, and
- adsorbate-freezing point.

This makes sense as all of the three original datasets (MOF features, adsorbate features, rigid uptakes) contribute to the remaining features of the LASSO-multi-linear regression model. These features are also useful features in common molecular simulations. This result may be a good indicator for feature selection for future work when larger datasets need to be processed.

5. Conclusions and future work

All the five regression models show good performance on prediction single-component adsorption uptakes for given (MOF, adsorbate) pairs near saturation pressure, and show the potential to be applied in the related research projects. Among these models, the multi-linear models (including the multi-linear regression model and the LASSO-multi-linear model) shares close values of r^2 with kernel-based models and MLPRegressor models, while are much cheaper to train and has relatively low complexity. The KRR model has the best prediction ability in terms of both r^2 and MEA. Therefore, multi-linear models and KRR may be the right tools to predict the single-component uptake considering the flexibility of the MOF framework.

For future work, we plan to expand the system of interest to multiple-component uptakes considering the MOF flexibility, where there are more features to take into account (e.g. compounds of multiple components, ratios of components) and the mechanism of adsorption may change from the single-component ones, where we would expect nonlinearity in the system. We expect that for the more complex system the kernel-based models and the MLPRegressor models may have better performance by capturing the nonlinear nature of the system.

Appendix A Individual contributions

	Chao-Wen Chang	Pengfei Cheng	Po-Wei Huang	Xiaohan Yu	Yamin Zhang
Data processing		o			
Baseline models			o		o
Improved models	o	o		o	
Model comparison					
r^2			o		o
training cost		o			
BIC		o			
kernel-based models comparison				o	
linear models comparison	o				
Presentation	o	o	o	o	o