Machine Learning for Heterogeneous Catalyst Selection in the Oxidative Coupling of Methane

Zhiren Bao
Civil and Environmental Engineering
zbao1995@stanford.edu

Jilong Lin
Civil and Environmental Engineering
jl64985@stanford.edu

Xinjing Xu
Management Science & Engineering
xxu346@stanford.edu

I. Introduction

Catalysts are pervasive in industries ranging from oil and gas to consumer products because they speed up reactions by providing lower-energy pathways. However, catalytic reactions are incredibly complex. Even for well studied reactions, the development of new heterogeneous catalysts remains very much an inefficient and empirical science due to the complexity of surface chemistry involved[1]. As such, many of the advances in this area have arisen from trial-and-error investigations. This empirical approach has uncovered useful information about catalysts on multiple length-scales; namely, from macro level features such as operating conditions to atomic level features such as elemental compositions and kinetics. To provide guiding insights for and improve efficiency of experiments, we leverage machine learning to facilitate the process of catalyst selection and development.

In this work, we study the oxidative coupling of methane(OCM), a chemical reaction that converts methane to more valuable C_2 products, such as C_2H_4 and C_2H_6[2]. Discovered by Keller and Bhasin in 1982[3], the reaction can for the major (desired) product can be described as:

$$2CH_4 + O_2 \longrightarrow C_2H_4 + 2H_2O$$

One of the plenty practical importance of this reaction is that the major desired product, C_2H_4, has numerous industrial applications. For example, it is the monomer of polyethylene and an important component of some logistics fuels[4]. Despite the economic promise, numerous obstacles, including catalyst selection, prevented this process from being effectively and efficiently used by industries. Therefore, we aim to develop a model that can predict the reaction yield given pertinent information about any given catalyst and the respective operating conditions. We performed data wrangling, feature engineering, and applied linear and nonlinear algorithms of different complexity.

Concretely, the input to our algorithms includes elemental information of the catalysts(e.g. molar composition, enthalpy of fusion, electronegativity), process parameters(e.g. operating temperature and pressure, residence time), catalyst preparation methods(e.g. precipitation, impregnation). We then use linear regression, ridge regression, Gaussian process regression, neural networks, and random forest to output a predicted yield of C_2H_4.

II. Related Works

Related works can be grouped into two categories: general approaches to use ML in assisting catalyst design, and incorporating ML to a specific reaction (OCM in our project).

Norskov et al. (2009) and Kitchin (2018) used ML to help understand catalysts in general[5, 6]. This general approach offers a broader view of ML in catalysis and is applicable to more reactions. However, because of its generality, it is not helpful with the actual design of catalysts for a single reaction.

Four papers used models and ML to study catalysts suitable for only the OCM reaction[1, 4, 7, 8]. Toyao et al. (2020) improved conventional ML algorithms on the OCM reaction and combined lab work with ML analysis[1]. Karakaya et al. (2017) focused on one pair of catalysts on the OCM reaction and used model-based interpretations to study one restriction which prevented OCM catalysts from being commercially viable[4]. Zavyalova et al. (2011) not only created a OCM catalyst database, but also found common features important for a high yield[7]. Suzuki et al. (2019) tried different algorithms on the same OCM reaction dataset that we used[8]. These papers which focused on only applying ML to the OCM reaction provided more detailed information such that if promising, the results could be directly used by industries. However, this approach has issues related with lack of compositional overlaps, bias from prior published data and low sample counts for many samples.

Our work used the similar approach as Suzuki et al. (2019). The authors tried algorithms including Lasso Regression, Ridge Regression, Kernel Ridge Regression, Support Vector Regression, Random Forest Regression, Improving Estimated Time to Restoration and XGBoost[8]. The algorithms we used will be discuessed in the next sections. We think that one possible reason that Suzuki et al. (2019) did not produce desirable outcomes is because of lack of features. Although the dataset consists of a number of important features, catalytic reactions are very complicated and features such as electronegativity, ionization energy, atomic weight etc. should not be ignored.

III. Dataset and Features

Our main database consists of 1866 examples each with 157 features (broad feature categories are shown in Table 1.), including cations and concentrations, anions and concentration, operating temperature and pressure, etc. As we aim to predict the C_2 yield for potential catalyst candidates, the label has a

dimension of 1866×1 (a distribution of the empirical yields is shown in figure 1). The raw data[8] was imported as a .csv file using python pandas module for pre-processing.

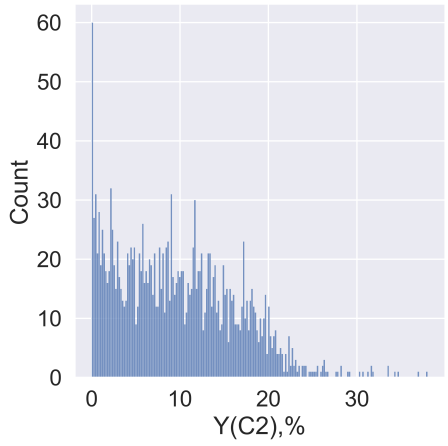

Fig. 1. Histogram of true yield

We performed one-hot encoding for non-numeric features such as elemental composition of the catalyst, support element types, and preparation methods. For features containing chemical elements, their encoded values are their molar compositions in that specific example. After one-hot encoding, we augmented our data set by adding atomic features that are unique to elements in the catalyst (index 5 in table I) from Toyao et al. (2020)[1].

A List of All Types of Features	
Index	Feature Name
1	Mole fraction of each catalyst (65 catalysts total) in each experiment
2	Mole fraction of each support element (18 support elements total) in each experiment
3	Mole fraction of each promotor (6 promotors) in each experiment
4	6 preparation types
5	Basic information of each element (atomic number, atomic weight, electronegativity, group, period, melting point, boiling point, enthalpy of fusion, density, ionization energy, number of valence electrons, surface energy)
6	Temperature
7	Partial pressure of CH_4
8	Partial pressure of O_2
9	Ratio of partial pressure $\frac{p(CH_4)}{p(O_2)}$
10	Total pressure
11	Contact time

TABLE I: Features

Implementation-wise, we utilized machine learning modules in Scikit-learn to perform ML analysis. For train-test splits, we selected 10% (186 samples) of the total entries as the test set, and the rest 90% (1680 samples) was used to train models.

IV. Methods

A. Evaluation Metric

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2 \tag{1}$$

We chose mean square error (MSE) as the error metric to compare predicted yield and true yield where m is the number of training examples, $\hat{y}$ denotes the predicted yield, and y denotes the true yield. We selected MSE as our metric because it is popular for supervised learning algorithms. The convexity of the function also makes it easier to optimize comparing to other metrics.

B. Linear Models

*1) **Linear Regression**:*

$$J(w) = \sum_{i=1}^{m}(y_i - (w * x^{(i)} + b)))^2 \tag{2}$$

$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}J(w) \tag{3}$$

To train linear regression, we minimize the lost function J(w) in equation 2. Specifically, we use equation 3 as the update rule for batch gradient descent, where j = 0,...,d and α is the learning rate. For every example, the algorithm runs through the whole dataset and updates w_j, until convergence. The predicted label, $w * x^{(i)} + b$ in equation 2, and the true label $y^{(i)}$ are used to calculate the cost function of this algorithm.

*2) **Ridge Regression**:*

$$J(w) = \sum_{i=1}^{m}(y^{(i)} - (w * x^{(i)} + b))^2 + \lambda\|w\|_2^2$$

To avoid over-fitting and to establish a baseline with reduced model complexity, we tried ridge regression. Similar with linear regression, ridge regression minimizes squared error and has the same update rule (equation 3), except the addition of a regularization parameter (λ) for the norm of the weights ($\|w\|_2^2$).

*3) **Gaussian Process Regression**:*

$$f(x) \sim GP(m(x), k(x, x')) \tag{4}$$

$$\begin{bmatrix} f(X) \\ f(X^*) \end{bmatrix} \sim N\left(\begin{bmatrix} \bar{m}(X) \\ \bar{m}(X^*) \end{bmatrix}, \begin{bmatrix} K(X,X) & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*) \end{bmatrix}\right) \tag{5}$$

$$\mathbb{E}[f(X^*)] = \bar{f}(X^*) + K(X^*,X)\left[K(X,X)\right]^{-1}\left(f(X) - \bar{\bar{f}}(X)\right) \tag{6}$$

Gaussian process regression (GPR) is a nonparametric, Bayesian approach to regression. We used Gaussian process regression since it works well on small datasets and has the ability to provide uncertainty measurements on the predictions. It also works well with kernel methods.

GPR specifies a prior distribution $p(w)$ by Bayes rule. Then a Gaussian Process Prior is assumed in equation 4. From

the Gaussian process prior, the collection of training points and test points are joint multivariate Gaussian distributed, as described in equation 5. The predictions is the $\mathbb{E}[f(X)]$ term in equation 6.

C. Non-linear Models

*1) **Neural Networks**:* Neural networks (NN) refer to broad type of non-linear models that involve combinations of matrix multiplication that other entry-wise non-linear operations (some non-linear activation functions include ReLU and Sigmoid). For one particular layer of NN, there are "neurons" that have trainable weights assigned to them; concretely, the prediction process(forward propagation) rule can be defined as

$$\begin{aligned} z^{[i]} &= W^{[i]}x^{[i-1]} + b^{[i]} \\ a^{[i]} &= f(z^{[i]}) \end{aligned}$$

where $W^{[i]}$ is the weights assigned to each neuron in the $i-th$ layer, $x^{[i-1]}$ is the input data from the $(i-1)-th$ layer, $b^{[i]}$ is the bias for the $i-th$ layer, f denotes some activation function, and $a^{[i]}$ is the activation for the next layer. To train a NN and update weights assigned to each neuron, back propagation with gradient descent (or other more complex algorithms such as Adam) is used.

*2) **Random Forest Regression**:*

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x') \tag{7}$$

Random Forest Regression is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. As an ensemble technique, it is capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. It has the capability to handle large data sets with large dimensions.In addition, It has methods for balancing errors in data sets where classes are imbalanced.

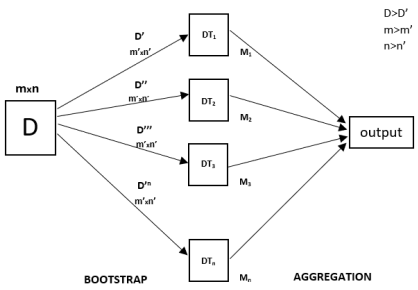

Fig. 2. Random Forest

The first step is bootstrap/bagging. For $b = 1, ..., B$ times, the algorithm samples n training samples X and Y, which are then used to train a regression tree classifier f_b. The second step is aggregation as shown in figure 2. After training, equation 7 is used to predict unseen samples x' by averaging the predictions from all the individual regression trees on x'. Random forests algorithm also uses a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is also called "feature bagging".

D. Cross Validation

As we have a relatively small dataset, we incorporate k-fold cross validation to assess the predictive performance of the models. If we set static training and test dataset, we can hardly notice whether all the data is independent. We do not know the level of noise influence either. K-fold cross validation can lower the bias and variance of our outcomes. Another benefit of k-fold cross validation is that all observations are used for both training and test. With k-fold cross validation, we can avoid some potential underfitting and overfitting problems. In our project, we split our dataset into 10 folds when training the models.

V. Results and Discussions

Intuitively, we believe nonlinear models will perform better than linear models because catalytic reactions, like most physical processes, are nonlinear in nature. Furthermore, we hypothesize that neural networks will perform the best with optimal network architecture and hyper-parameters because of their immense potential to fit complex functions. However, we found that random forest performed the best. A summary of the results can be found in table II.

A Summary of Results			
Model	Train Error	Test Error	R^2
Linear Regression	24.46	34.63	0.33
Ridge Regression	25.99	21.13	0.44
Gaussian Process Regression	27.20	26.46	0.36
Neural Network	25.43	32.14	0.26
Random Forest	2.47	2.55	0.94

TABLE II: Result Summary

A. Linear Models

We tried several linear regression models, including Linear Regression, Ridge Regression, and Gaussian Process Regression. The coefficient of determinations for these three models are 0.33, 0,44, and 0.36, respectively. Apparently, ridge regression models has the best performance among three models; however, the $R^2 = 0.44$ is not persuasive in this case. MSEs of Linear Regression model, Ridge Regression model, and Gaussian Process Regression model are 34.63, 21.13, and 26.46, respectively. Figures 3, 4, 5 shows the relationship between prediction and actual results for three models. In our ridge regression model, we set the learning rate α as 1.

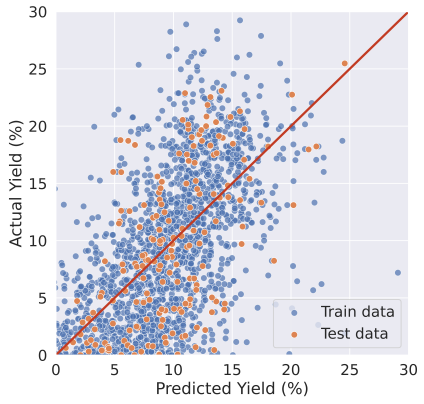

Fig. 3. Linear Regression $R^2 = 0.33$

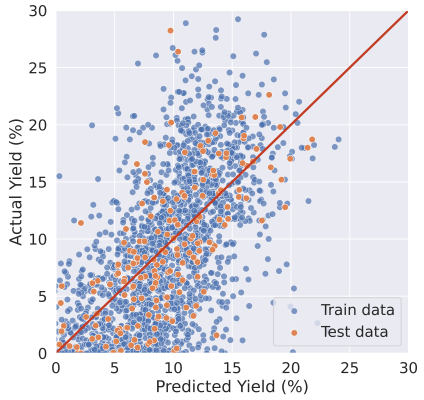

Fig. 4. Ridge Regression. $R^2 = 0.44$

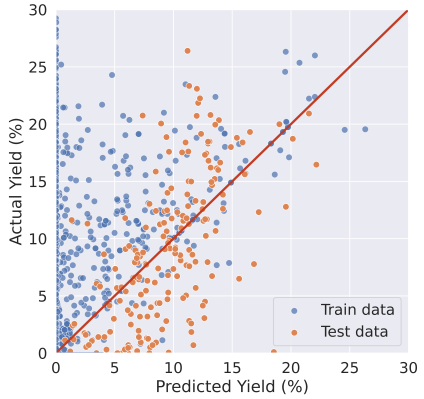

Fig. 5. Gaussian Process Regression. $R^2 = 0.36$

This figure shows that the predictions are relatively off, with the R^2 scores of three linear regression models all below 0.5 and large MSEs. This is because linear models are too simple to model catalytic reactions. In addition, we have only 1866 data points, which is a small dataset. A small dataset tends to have underfitting problem, and this could occur in our case.

B. Non-Linear Models

1. Neural Network

After extensive hyperparameter search with the objective of minimizing 10-fold cross-validated MSE, we used a fully-connect neural network with five hidden layers (size=(150, 128, 64, 32, 46)), an initial learning rate of 0.01, a regularization strength of 0.5, and the Adam optimizer. The MSE for neural networks is 25.43 and 32.14 for train and test sets, respectively. A test error of 32.14 means that the hyperparameters we chose to optimize did not achieve desired outcomes. The algorithm not only didn't fit the data as well as we expected, but also didn't generalize to the test data (since the test and train errors are different by 7.)

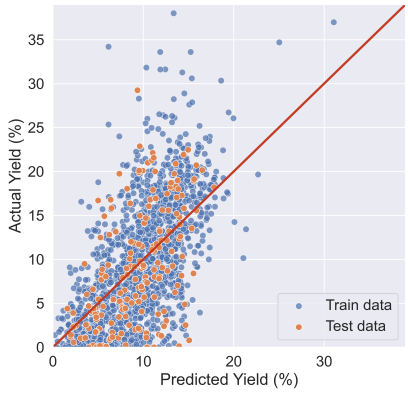

Fig. 6. Neural Network. $R^2 = 0.26$

2. Random Forest

For hyperparameter tuning, we consider four parameters: min sample leaves, min samples split, n estimators and max depth. We use training data to tune these four parameters. The lowest error is always achieved when min sample leaves and min samples split take the default values, 1 and 2 respectively. Since each time the lowest error is achieved with different values of n estimators and max depth, we make a list for possible best values for n estimators and max depth. We use a nested loop to find the best combination of the two parameters which gives the lowest error. Using the best performed parameters, we fit the training set and test set separately and find MSE for each. As the model with the best performance, the training error is only 2.47, and the test error is 2.55. This means that the algorithm performs a little better on the training set than on the test set. In addition, the R^2 scores is 0.94, which outperforms any other models. Originally, we thought Neural Network would perform the best, but Random Forest becomes the winner ultimately.

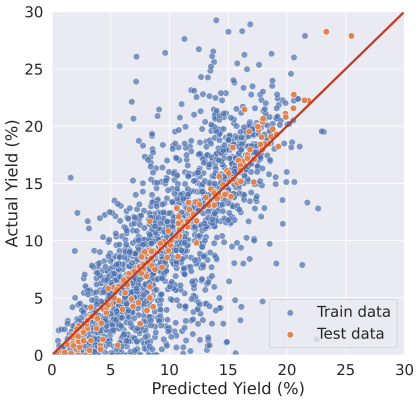

Fig. 7. Random Forest Regression. $R^2 = 0.94$

VI. Conclusions & Next Steps...

Machine learning is proven to be able to predict OCM yield as our highest coefficient of determination value can reach 0.94. The comprehensive data analysis from the previous OCM experiments' data can help researchers design the catalytic material in the future. With the models that we conduct for yield predictions, researchers can utilize the weights, in particularly the weights from Random Forest Regression model, to estimate the potential yield of a future catalytic reaction.

There are many routes for future work. Firstly, researchers could use principle components analysis to identify which features are the most important contributing factor to product yield; this would provide researchers with unique insights to design a high-performance catalyst. Secondly, the OCM reaction mechanism could be used as features; we could approximate catalyst-specific activation energies to improve the accuracy of the prediction. Lastly, we believe that we could introduce the molecular geometry of different catalysts to capture the diffusion process for the product and the reactants; with these spatial features, convolutional neural network could help with picking up nuances of different geometries which are not captured by fully connected neural networks.

VII. Individual Contributions

Every member in the team contributed equally to this project. Each member worked on data pre-processing, tuning algorithms, and writing this report.

VIII. Code

Source code for the project can be found here: https://github.com/xuxinjing/cs229_catalysis.git

References

[1] Takashi Toyao, Zen Maeno, Satoru Takakusagi, Takashi Kamachi, Ichigaku Takigawa, and Ken-ichi Shimizu. Machine Learning for Catalysis Informatics: Recent Applications and Prospects. *ACS Catalysis*, 10(3):2260–2297, 2019.

[2] Qijian Zhang, Dehua He, and Qiming Zhu. Recent progress in direct partial oxidation of methane to methanol. *Journal of Natural Gas Chemistry*, 12(2):81–89, 2003.

[3] G.E. Keller and M.M. Bhasin. Synthesis of ethylene via oxidative coupling of methane: I. determination of active catalysts. *Journal of Catalysis*, 73(1):9 – 19, 1982.

[4] Canan Karakaya, Huayang Zhu, Bahman Zohour, Selim Senkan, and Robert J. Kee. Detailed Reaction Mechanisms for the Oxidative Coupling of Methane over La2O3/CeO2 Nanofiber Fabric Catalysts. *ChemCatChem*, 9(24):4538–4551, 2017.

[5] Jens Kehlet Nørskov, Thomas Bligaard, Jan Rossmeisl, and Claus Hviid Christensen. Towards the computational design of solid catalysts. *Nature chemistry*, 1(1):37–46, 2009.

[6] John R Kitchin. Machine learning in catalysis. *Nature Catalysis*, 1(4):230–232, 2018.

[7] Ulyana Zavyalova, Martin Holena, Robert Schlögl, and Manfred Baerns. Statistical Analysis of Past Catalytic Data on Oxidative Methane Coupling for New Insights into the Composition of High-Performance Catalysts. *ChemCatChem*, 3(12):1935–1947, 2011.

[8] Keisuke Suzuki, Takashi Toyao, Zen Maeno, Satoru Takakusagi, Ken-ichi Shimizu, and Ichigaku Takigawa. Statistical Analysis and Discovery of Heterogeneous Catalysts Based on Machine Learning from Diverse Published Data. *ChemCatChem*, 11(18):4537–4547, 2019.