CHEME 6880 | Final Project Report

Using supervised learning methods to predict Buchwald-Hartwig amination yields from high throughput experimentation data

Nathan M. Lui

Abstract

Owing to the challenges surrounding molecular representation the field of chemistry has arrived late to the party, nevertheless machine learning methods are on the rise in chemical research, particularly in tandem with the development of high throughput and continuous flow chemistry. In this study we examine the Buchwald-Hartwig cross coupling high throughput screening set created by Doyle and coworkers as a tool for the prediction of C-N cross coupling yields in the presence of isoxazole additives. We trained 13 different models across five classes of machine learning algorithms, evaluating these on a hidden test set of four additives. We show that a 5-nearest neighbors model provides significantly better performance (RMSE: 12.3%) than other regression schemes as well as the one-hot encoded baseline models.

Introduction

The world is made of chemistry; from the materials that form your everyday electronics, to the petroleum products that power the global supply chain, to the hundreds of thousands of pharmaceuticals that are prescribed by doctors every day across the world, molecules are an inextricable part of our lives. A lucky handful of these compounds (e.g., latex, crude oil, food, etc.) are created by mother nature herself; for everything else (e.g., synthetic plastics, industrial dyes, most medicines, etc.) there's a large-scale manufacturing process. The more complex the compound the more complicated the process. This aphorism is exemplified nowhere better than in the manufacturing of small-molecule active pharmaceutical ingredients (APIs).

Pharmaceuticals are typically small, but incredibly complex molecules. Designing large-scale syntheses for small molecules requires efficient and incredibly robust chemical reactions. Consider a 10-step plant-scale process for a particular pharmaceutical drug. If every step of that process causes just 5% loss (95% efficiency), over 40% of the product will be lost to production. Given the cost of chemical feedstocks and potential losses to purification and transport it becomes clear that industrial process chemistry requires nearly perfect reactions. The *academic* synthetic chemistry community has begun to understand this fact and in the design of more efficient transformations has turned to machine learning methods for the optimization of reaction conditions and even the design of reagents and new reactions.[1–4]

Synthesis of aryl- and heteroaryl-amines

There has been considerable research effort directed at the development of synthetic routes to aryl- and heteroaryl-amines (**Chart 1**) because these are important building blocks for pharmaceuticals, agrochemicals, organic dyes, and functional polymers, etc.[5] A glance at the world's best-selling pharmaceuticals from the past several years reveals the importance of this structural motif (**Chart 2**). Prior to 1995 the best way of forming this structure without highly toxic reagents was a fairly expensive reaction with limited substrate scope called the S_NAr. Today, while the S_NAr remains a significant tool for aryl C-N bond formation it has been largely replaced in the academic synthetic community by a reaction known as the Buchwald-Hartwig amination.

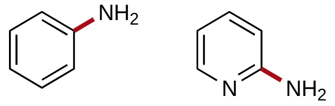


Chart 1 | Aniline (left) and 2-aminopyridine (right), the simplest aryl- and heteroaryl-amine, respectively. The carbon-nitrogen (C-N) bond formed from the Buchwald-Hartwig amination.

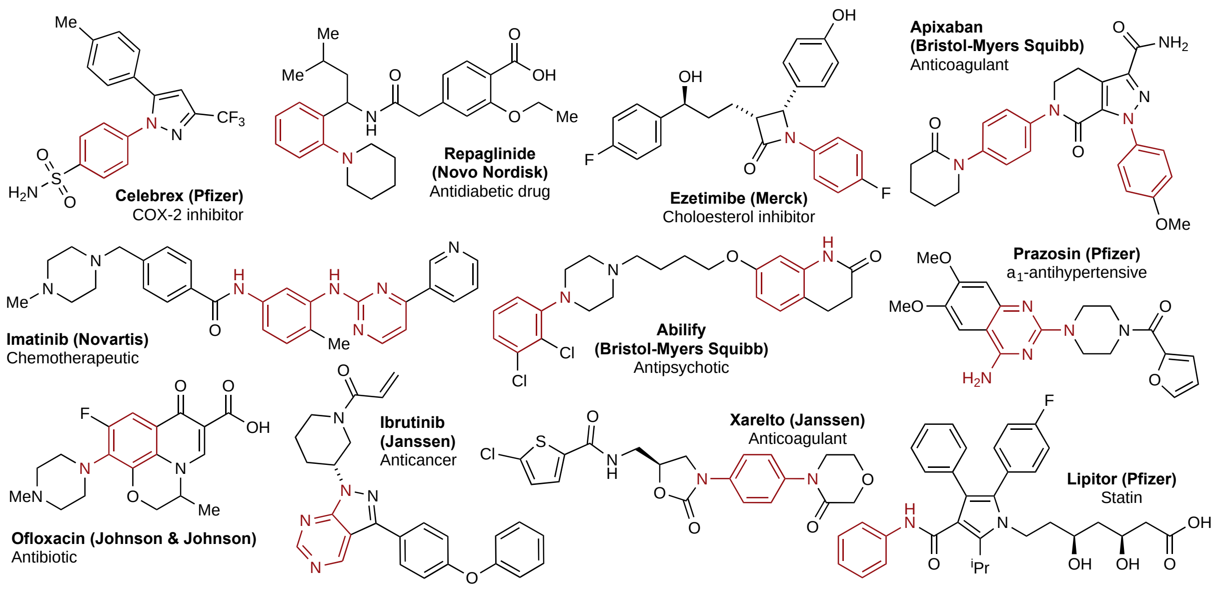
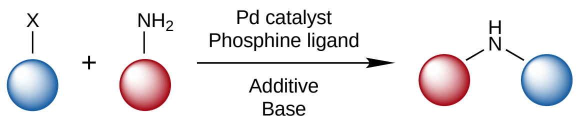


Chart 2 | Examples of top-selling active pharmaceutical ingredients (APIs) containing the arylamine and herteroaryl motif (substructure highlighted in red). A list of chemical abbreviations is included after the references.

The Buchwald-Hartwig amination

The seminal reports of what would become the Buchwald-Hartwig amination was first published by Migita and coworkers who developed the palladium (Pd)-catalyzed substitution of aryl halides using incredibly toxic amino-tin reagents.[6] In 1995, a decade later the groups of Stephen Buchwald (MIT) and John Hartwig (Berkeley) simultaneous published tin-free Pd-catalyzed couplings of aryl halides (blue) and amines (red) (**Scheme 1**).[7] In the twenty years since, what has come to be known as the Buchwald-Hartwig C-N cross-coupling (because the reaction generates the carbon-nitrogen (C-N) bond shown in red, **Chart 1**) or the Buchwald-Hartwig amination, has become one of the most widely used organometallic reactions in synthetic chemistry.[7,8] Owing to the utility of the Buchwald-Hartwig reaction a veritable library of ligands, bases, and additives have been developed to facilitate reactions between almost every possible aryl halide and amine (**Chart 3**).



Scheme 1 | General model of the palladium (Pd) catalyzed Buchwald-Hartwig C-N cross coupling reaction.[8]

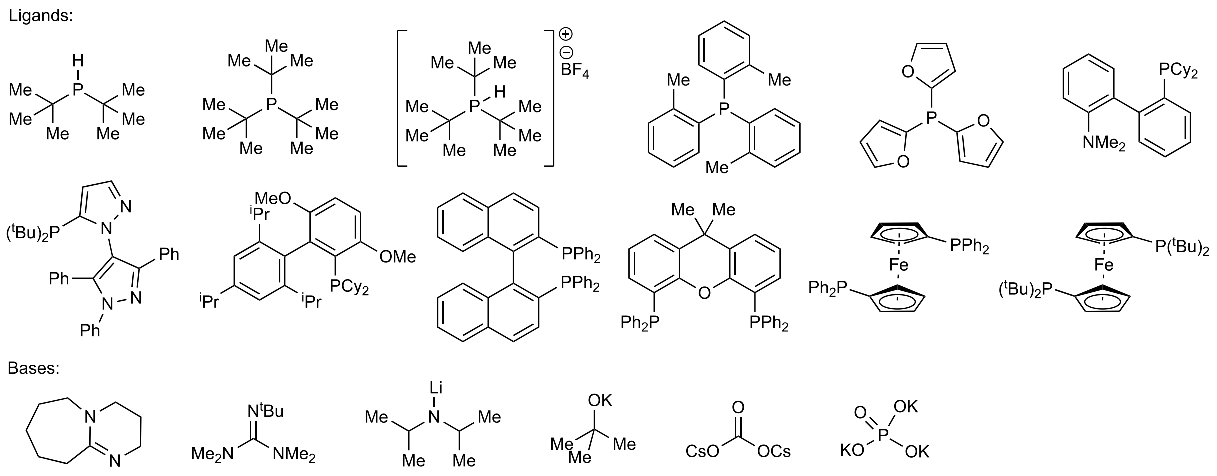


Chart 3 | Examples of commercially available phosphine ligands and bases used for Buchwald-Hartwig C-N cross coupling.[8] A list of chemical abbreviations is included after the references.

Previous work

In 2018, the group of Prof. Abigail Doyle (Princeton) set out to further optimize the Buchwald-Hartwig amination.[1] They noticed the poor performance of the cross-coupling reaction in the presence of more complex "drug-like" molecules, especially those containing 5-membered 1,2-heterocycles (see additives in **Figure 1** in the next section). In collaboration with Merck Research Labs, they screened over 4000 reaction combinations generating models to determine the effects of these additives on the reaction yield. Their model and other efforts to predict reaction yields (both using Doyle's Buchwald-Hartwig dataset and other common reaction databases) is described in later sections. In this project we attempt to improve on the model determined by Doyle and coworkers on the Buchwald-Hartwig dataset, by incorporating not only the original computed features, but also high- and low-level static representations of the molecules.

Background

Choice of dataset

The original Buchwald-Hartwig amination dataset developed by Doyle and coworkers in collaboration with the high throughput screening facility at Merck Research Labs consists of over 4600 reactions (**Figure 1a**).[1] The published dataset maps five compounds (a palladium catalyst, a base, an aryl halide, an isoxazole additive, and a coupled product) to the percent yield of the Buchwald-Hartwig amination. These reactions were run simultaneously on several 1536-well plates under identical nanomolar reaction conditions. In total 4 catalysts, 3 bases, 15 aryl halides, and 23 isoxazoles were screened (**Figure 1b**).[1] Approximately 30% of the reactions failed to produce any product, however the remaining 70% are relatively uniformly distributed (**Figure 2**).[1]

Doyle's Buchwald-Hartwig dataset was chosen because it is chemically complete. The sampled reaction space is fully covered by the dataset. Every aryl halide is screened with every palladium catalyst, every additive, and every base, giving us a full view of the sampled reaction landscape. One must note that just because the sample space is fully covered does not imply that the *global reaction space* is adequately covered. Generalizability to *out-of-set* reactions (i.e., Buchwald-Hartwig aminations involving coupling partners that are *not* studied) is, of course, largely dependent on their similarity to the compounds sampled. Nevertheless, the likelihood of successful modeling of the *in-set* reaction landscape (and yield prediction) is greatly improved by the completeness of the dataset. Note that a complete sample space necessitates special generalization test set treatment; this will be further discussed in the methods section below.[2,4,9]

Other well-explored reaction datasets have been used to predict reaction yields; two of the most popular are the USPTO and its subset USPTO-50k, which contain 424,621 and 50,036 (respectively) atom mapped reactions from organic chemistry patents from the US Patent and Trademark Office.[10] While these would be more than suitable for modeling reaction yields, their vast generality (over 10 general reaction classes) makes the modeling task much more complicated. Furthermore, the USPTO datasets are mined from published patents resulting in approximately 70% accuracy in product yields and 89% accuracy in chemical identity.[10] These factors lead us to use the Buchwald-Hartwig dataset despite its smaller size and scope.

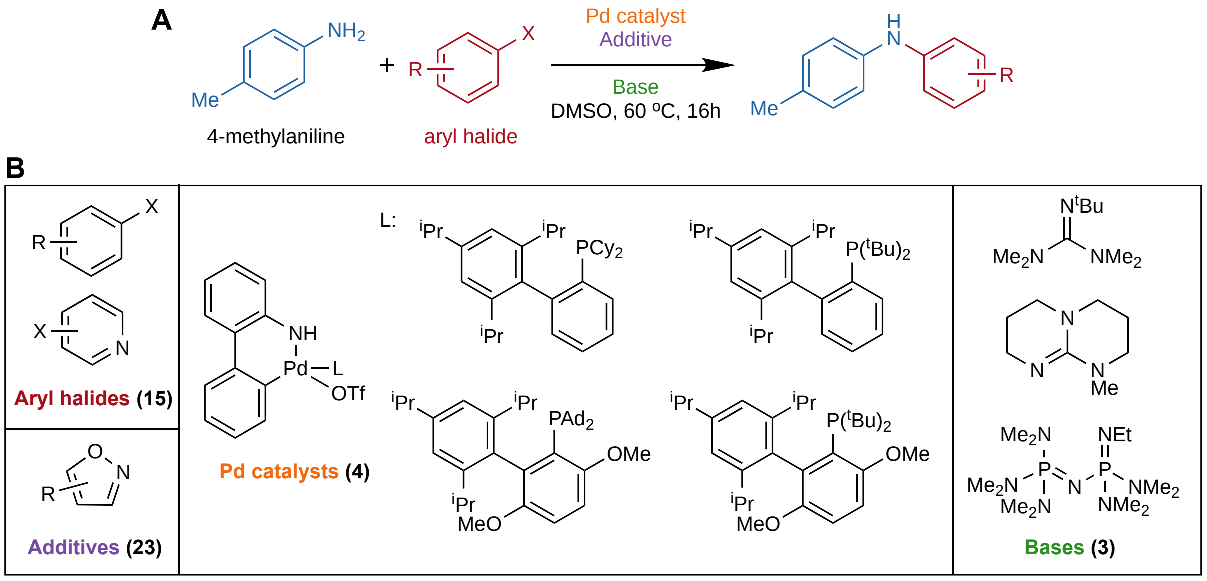


Figure 1 | (a) The Buchwald Hartwig amination screened in the dataset. (b) Four chemical variables (15 aryl halides, 4 palladium catalysts, 3 bases, and 23 isoxazole additives) were studied simultaneously using high throughput screening in 1536-well reaction plates. Each well was run under the same conditions (nanomolar-scale) and the yield of each combination of reactions was determined spectroscopically. Figure adapted from Ahneman et al.[1]

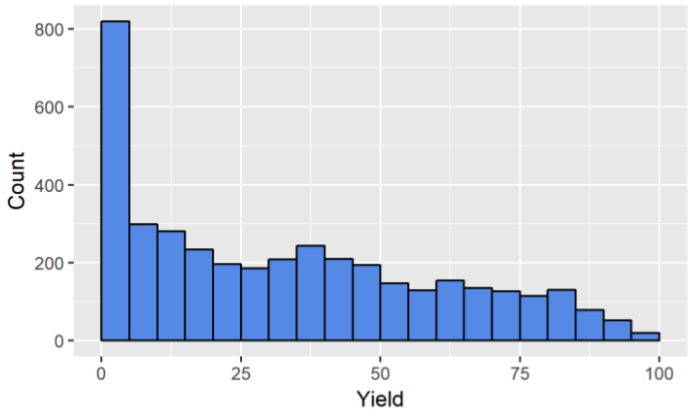


Figure 2 | The distribution of reaction yields (labels) from Doyle's Buchwald-Hartwig dataset. The control reactions have been removed. Approximately 30% of the reactions failed to produce any product, however the remaining 70% are relatively uniformly distributed. Figure reproduced from Ahneman et al.[1]

The challenge of molecular representation

One of the largest obstacles to the application of machine learning methods to chemistry is the conversion of molecules to meaningful input data. A sufficiently learned student can be taught general rules/models to characterize the differences in, for example, the three bases or four catalysts in the dataset, however the rules of undergraduate organic chemistry can hardly be used to translate complex 3-dimensional molecules into machine-readable features. The question of how one translates molecules to feature vectors began with quantitative structure–activity relationship (QSAR) and quantitative structure–property relationship (QSPR) modeling in the pharmaceutical industry where QSAR modeling has been used to screen libraries of compounds for "druglikeness," reducing the possible chemical space by orders of magnitude.[2,11] In these QSAR/QSPR studies important molecular descriptors such as a compound's molecular weight, dipole moment, solubility, or "linearity" are computed for each molecule in the dataset.[12] There have been recent efforts to apply the methods of natural language processing and graph theory to chemical structures; while these methods have shown considerable promise they are still in their infancy.[2,9,11,13] The "computable descriptor" method is the workhorse of modern chemical machine learning.

In the original paper Doyle and coworkers[1] used density function theory (DFT) methods to generate features for the 46 compounds. A total of 120 various chemical descriptor (e.g., intramolecular vibrational modes/intensities, dipole moment, electrostatic charges, etc.) were calculated for the compounds (see supplementary information). These data are included in the original dataset, bringing the total size of the dataset to 4140 samples in $\mathbb{R}^{120}$. It would be interesting for further study to consider the impact of additional molecular descriptors on the analysis presented here, however the wide range of molecular descriptors available and the challenges surrounding the combination of calculated features and multi-dimensional molecular fingerprints in a balanced manner prevented the original plan of inclusion into the original Ahneman dataset.[14]

Methods

Preprocessing

Preprocessing the data (e.g. removal of correlated features, outlier analysis, etc.…) was largely unnecessary due to much the work already done by Ahneman et al.[1] however a small change to the dataset was made. In the additive category the "none" entries were represented by **0**; this featurization is chemically sound as the molecular properties (such as electrostatic charge, dipole moment, electronegativity, volume, weight, etc.…) for "no molecule" are all 0. All models were trained on the dataset with and without normalization to 0 mean and unit variance.

Model screening

The roughly twenty models examined in this study fall into five different classes: linear models, support vector-based model, nearest neighbor models, tree-based models, and neural network models. Within these classes, in particular the linear and tree-based models, various configurations, regularization schemes, and ensembling methods were explored as well. In the class of multivariate linear regression, the common ℓ_1 (LASSO) and ℓ_2 (ridge) regularization schemes were explored alongside the combination elastic net regularization, and generalized linear models (also known as Tweedie models). Within tree-based models, random forest models,[15] gradient boosted regression trees,[16] adaboost trees,[17] and XGBoosted trees were examined in addition to the original CART model. Model performance was evaluated on a hidden test set in order to establish models' generalizability to out-of-sample models (*vide infra*).

Model evaluation and holdout testing

Candidate models are trained for generalizability using a hidden test set. Typically, the dataset is split randomly into a train and test set (in their original study Ahneman et al.[1] utilized a random 70%/30% split). The model is trained using only the training set samples and then evaluated on the test set to determine how well the model generalizes to unseen data.[18] However, random splitting of a combinatorially complete dataset creates a training set in which some combinations of molecules are hidden, but *every* molecule is eventually seen by the learning algorithm.[4,9] Random sampling of the test set allows it's members to infiltrate the training set, reducing the problem at hand to one of simple data imputation, and producing overly optimistic metrics of generalizability.

To avoid this effect the candidate models were evaluated on a hidden set of four randomly selected additives (**Figure 3**). The additives were chosen to split the dataset upon because they are the largest and most chemically diverse set of compounds. There are 22 different additives with a wide range of chemical functionality, allowing us to split the dataset into sufficiently different testing and training sets. After splitting 4 of the 22 additives off into the test set, we were left with 3420 training samples (an approximately 80/20 train/test split). We demonstrate in later sections the overly optimistic test errors resulting from purely random sampling of the test set.

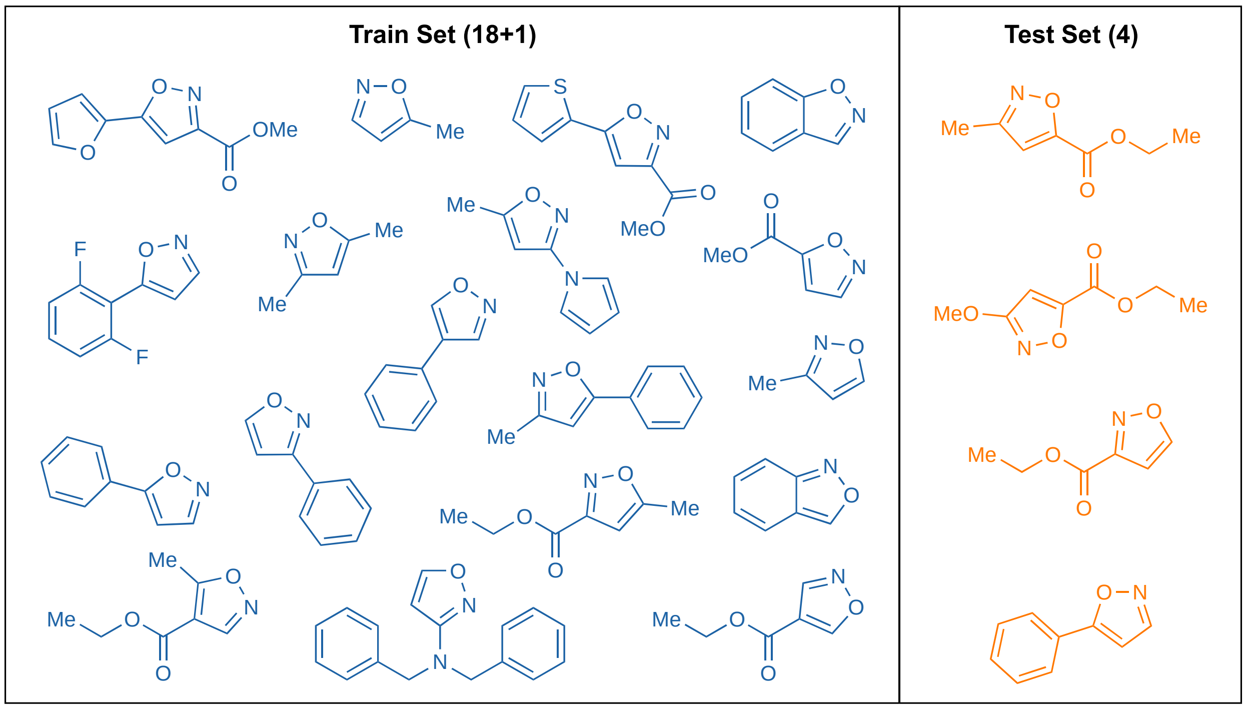


Figure 3 | Division of isoxazole additives into a randomly selected aware training set (18 molecules + no additive) and a hidden test set (4 molecules). The resulting train set is comprised of 3420 samples and the test set of 720 samples.

Hyperparameter tuning and cross validation

The selection of model hyperparameters is one of the most important decisions in machine learning.[19] Typically this process is accomplished through an extensive search of the hyperparameter space using cross validation. In traditional *k*-fold cross validation, the dataset is randomly divided into "folds" or subsamples and each hyperparameter combination is trained on *k*-1 of these folds and then tested on the holdout fold. This process is repeated for all *k* folds and the test fold errors (termed, cross validation errors) are averaged to provide an unbiased estimator of the mean training error. The full process is repeated for every combination of hyperparameters, and the model is then retrained on the full dataset using the set of hyperparameters with the lowest mean CV error.[18,19]

Many strategies of drawing cross validation folds exists including Leave-One/p-Out (LOOCV/LPOCV), random permutations (a.k.a. shuffle & split), stratified *k*-fold, and time-series split.[19,20] However many models of cross validation hinge, like much in the realm of machine learning, on the assumption that datapoints are drawn independently and identically distributed (commonly referred to as, *i.i.d.*) from the parent population one hope to model.[18] While this central assumption holds in many cases, it is particularly untrue in our situation. The dataset is built from high throughput screening data in which every compound in each of the four classes (additives, aryl halides, ligands, and bases) is screened with every compound in the other three classes. While this method provides a holistic view of the reaction space it very clearly violates the independence assumption.[4]

To remedy this we adopt the recommendations of many recent publications of the field and utilize leave-one-*molecule*-out cross validation (LOMOCV); formally this process is analogous to leave-*p*-out cross validation.[2,4,9,18] As in above, the folds are drawn among molecules in the additive category of the dataset (*vide supra*), providing the added benefit of allowing us to estimate *out-of-sample* predictive power. After partitioning off the test set and splitting the training set along the additive molecules (**Figure 3**) we arrive at 19 validation folds (18 different additives and "no additive") of 180 samples each.

Performance against baseline models

Generalizability of the candidate models was tested on a holdout test set following the methodology described above. In addition to test set evaluation, individual models were also tested against baseline models trained on a dataset of one-hot encoded vectors as described by Chuang and Keiser[9] and Żurański et al.[2] The baseline models were trained and tested following the same methodology as the feature encoded models. Outperformance of the one-hot baseline models ensures that the algorithm is taking into consideration molecular features instead of just "reading" the dataset. Additionally, where applicable, the performance of the best models in each class from Żurański et al.[2] are included as a benchmark.[a]

Results and discussion

Implementation

Data handling and processing was primarily completed using the NumPy and Pandas libraries and plots were made using matplotlib. Machine learning algorithms were implemented using the Scikit-Learn machine learning library.[20] Data normalization was accomplished using the using Scikit-Learn's StandardScaler class. For all but the regularized linear models (LASSO, ridge, and elastic net), cross validation for the selection of hyperparameters (both *k*-fold and leave-one-molecule-out) were implemented using the GridSearchCV class (with the KFold and LeaveOneGroupOut iterators, respectively) using the mean squared error scorer. For the linear models above, cross validation was implemented using their respective built-in validator classes (LassoCV, RidgeCV, ElasticNetCV). XGBoosted regression was implemented using the XGBoost Python package.[21]

Model performance

In this section models are described by class; from each class the best performing model is compared to that of Żurański et al.[2] as well as the one-hot vector encoded baseline model.[b] Ideally, the best performer from each class outperforms the one-hot baseline model, but underperforms relative to the "random split" model, in which the test set is randomly (in a non-stratified manner) drawn from the dataset. All models were trained on the dataset with and without normalization to 0 mean and unit variance. With the exception of tree-based methods, in which performance of scaled and unscaled models were identical, the scaled models performed better than the unscaled models in every case examined.

[a] The analysis in Żurański et al.[2] was conducted in R using the caret package, so to maintain comparability models were trained using the "optimal hyperparameters" in ref [2] and tested in the manner described above. Ref [2] also lists mean CV errors for several models (Fig. 3a in ref [2]), in all cases the best performing models presented in our analysis outperforms these mean CV errors as well.

[b] Performance metrics and optimized hyperparameters for all models are documented in supplementary information.

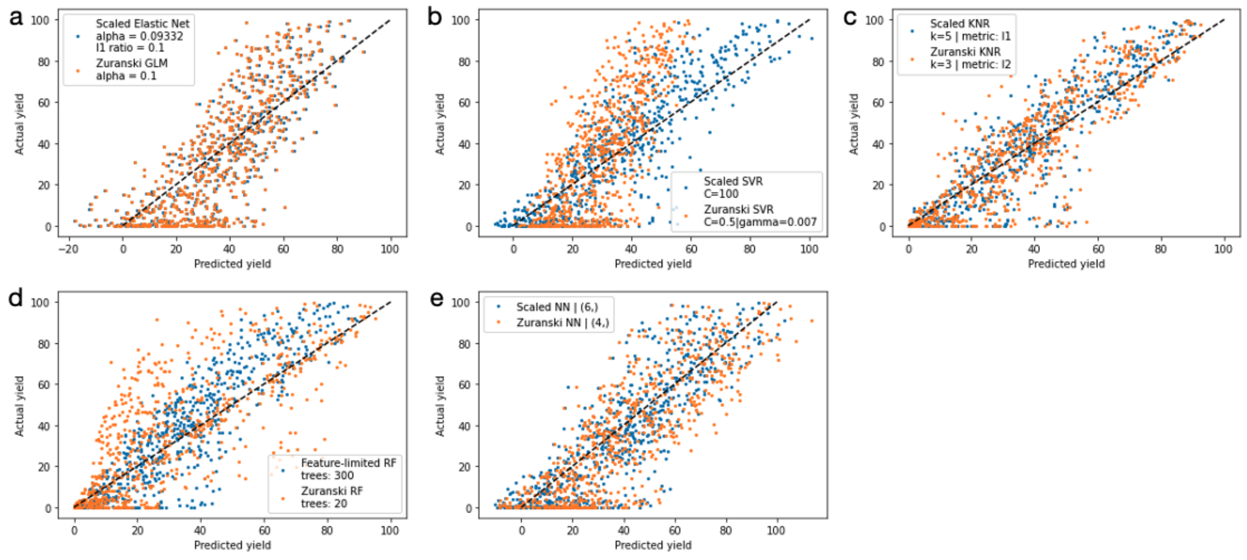


Figure 4 | Observed vs. predicted plots for the test set molecules of the best performers in each model class (blue) and the best model proposed by Żurański et al.[2] (orange). **a** | Best linear model (elastic net regression, $\alpha = 0.09332$, ℓ_1 ratio = 0.1, $R^2 = 0.61$) and Żurański model (generalized linear model, $\alpha = 0.1$, $R^2 = 0.61$). **b** | Best support vector regressor (C = 100, $R^2 = 0.78$) and Żurański model (C = 0.5, $\gamma = 0.007$, $R^2 = 0.40$). **c** | Best k-nearest neighbor regressor ($k = 5$, distance metric: ℓ_1, distance weighted, $R^2 = 0.82$) and Żurański model ($k = 3$, distance metric: ℓ_2, unweighted, $R^2 = 0.79$). **d** | Best tree-based regressor (random forest, trees = 300, feature-limited, $R^2 = 0.79$) and Żurański model (random forest, trees = 20, $R^2 = 0.55$). **e** | Best neural network (6 hidden nodes in a single layer, $R^2 = 0.72$) and Żurański model (4 hidden nodes in a single layer, $R^2 = 0.66$). Separated plots can be found in the supplementary information.

Linear models

Among the class of multivariate linear models, elastic net regression outperformed ridge regression, LASSO, ordinary least squares (OLS), and generalized linear models (**Table 1**). Iterative cross validation was used to tune, where applicable, both α, the regularization strength (LASSO, ridge, and elastic net), and the ℓ_1 ratio (elastic net). The optimal hyperparameters found for the elastic net regressor were an ℓ_1 regularization ratio of 0.1 and $\alpha = 0.09332$. Nevertheless, multivariate linear models, in general, performed the poorest of all the method examined. The best performing model trained on molecular performed just as well as that proposed by Żurański et al.[2] (**Figure 4a**) but neither are better than the best linear model trained on one-hot vectors suggesting poor generalizability and an inability to interpret the underlying model chemistry.

Table 1 | Test set performance of various linear models (Best: Elastic Net Regression) against the baseline one-hot vector and Żurański models.[2] The "Random Split" column is the optimistically biased model resulting from blind sampling of the test set.

	GLM	**LASSO Regression**	**Ridge Regression**	**Elastic Net Regression**	**One-Hot Model**	**Żurański Model**	**Random Split**
RMSE	18.23	18.19	17.87	17.91	17.15	17.93	13.05
$\boldsymbol{R^2}$	0.59	0.60	0.61	0.61	0.64	0.61	0.77

Support Vector Regression

Support vector regression (SVR) was explored as an alternative to simple linear models. Several rounds of grid search cross validation were used to explore the effects of various kernel functions (linear, gaussian, and sigmoidal) and regularization strengths (C = 0.0001 to 1000). The optimal model hyperparameters were a regularization strength C = 100 using a gaussian kernel. The support vector methods performed quite well compared to the both the baseline one-hot model and the Żurański model suggesting a deeper understanding of the underlying features by the model (**Figure 4b**, **Table 2**). Additionally, the better performance of the SVR models over the linear suggests that the underlying structure of the data is inherently nonlinear.

Table 2 | Test set performance of the optimized support vector regression model against the baseline one-hot vector and Żurański models.[2] The "Random Split" column is the optimistically-biased testing error resulting from blind sampling of the test set.

	Support Vector Regression	**One-Hot Model**	**Żurański Model**	**Random Split**
RMSE	13.64	14.20	22.26	11.58
$\boldsymbol{R^2}$	0.78	0.76	0.40	0.82

k-Nearest Neighbors

A *k*-nearest neighbors' model was the reasonable next step under the assumption that molecules of similar properties/identity would react similarly. Iterative grid search cross validation was conducted to explore the optimal size of the neighborhood (1 – 100) as well the distance metric (Minkowski distance: p = 1 or 2). The optimal parameters for the *k*-NN regressor were a neighborhood of 5 points using the ℓ_1 Minkowski distance. The final prediction was made as a distance-weighted average of the labels in the neighborhood. The *k*-NN regression models are the best performing models in this analysis. Our model outperforms not only the baseline one-hot and the Żurański models, but curiously also the randomly split model (**Figure 4c**, **Table 3**). Outperformance of the randomly split model suggests superior out-of-sample generalizability, most likely owing to overfitting by a model which has the ability to see every compound in the dataset (*vide supra*). This suggests that the model performance might be able to be tuned using the *number of molecules in the training set* to prevent overfitting, in a process somewhat akin to early stopping; this is, of course, a potential avenue to be explored in further studies.

Table 3 | Test set performance of the optimized *k*-nearest neighbor regression model (k = 5, ℓ_1–distance weighted) against the baseline one-hot vector and Żurański models.[2] The "Random Split" column is the optimistically-biased testing error resulting from blind sampling of the test set.

	***k*-NN Regression**	**One-Hot Model**	**Żurański Model**	**Random Split**
RMSE	12.26	15.21	13.24	15.29
$\boldsymbol{R^2}$	0.82	0.72	0.79	0.69

Tree-based models

Within the class of tree-based models we examined basic regression trees as well as various bagging and boosting methods. As expected, simple decision trees performed the worst of all tree-based models, even with depth limiting to a minimum of 300 samples per leaf. Boosting methods, namely Adaboost, GBRT, and XGBoost, all gave similarly poor performance (RMSE: 16.11 – 18.80). Successive rounds of cross validation as well as early stopping could not improve these models further. The random forest model with 300 trees and limited to 7 features outperformed the one-hot baseline model, the Żurański model, and all of the boosted tree methods. As expected, the randomly split model outperforms tree-based regressors trained on the molecule-stratified training set (**Table 4**). It is notable that the random forest model developed by Żurański et al.[2] (20 trees) was the best model (**Figure 4d**).

Table 4 | Test set performance of the bagged and boosted tree-based regression models against the baseline one-hot vector and Żurański models.[2] The "Random Split" column is the optimistically-biased testing error resulting from blind sampling of the test set.

	Gradient Boosted Regression	**AdaBoost Regression**	**XGBoost Regression**	**Random Forest Regression**	**One-Hot Model**	**Żurański Model**	**Random Split**
RMSE	16.11	18.80	17.14	13.17	14.37	19.30	6.73
$\boldsymbol{R^2}$	0.69	0.57	0.64	0.79	0.75	0.55	0.94

Neural Networks

Finally, simple neural networks were explored due to their good performance in Ahneman et al.[1] and Żurański et al.[2] however we were not so lucky. Using iterative cross validation to screen several sizes of hidden layer sizes and configurations (up to 100 neurons in 1 or 2 layers) and regularization strengths (α from 0.000001 to 0.0001) we found the optimal size of the neural network to be 6 hidden neurons in a single layer using the standard ReLU activation, similar to that of Żurański et al.[2] which employed a single layer neural network with 4 hidden neurons (**Figure 4e**). Neither our model nor the Żurański model managed to outperform the one-hot encoded baseline model suggesting deeper seeded issues at play (**Table 5**). This was unexpected as there exists an inherent nonlinearity that the neural network should be able to understand, however given the time constraints of the project we were unable to implement some of the more complex neural network architectures that have been previously successful at solving similar problems.[11,22,23] We leave the further development of the neural networks open for future study.

Table 5 | Test set performance of the optimized neural network regression model against the baseline one-hot vector and Żurański models.[2]

	Neural Network Regression	**One-Hot Model**	**Żurański Model**
RMSE	15.22	14.33	16.75
$\boldsymbol{R^2}$	0.72	0.75	0.66

Discussion

Of the top five models in their classes, only three outperformed the one-hot baseline models: the *k*-nearest neighbor regressor, the support vector regression model, and the random forest regression model. It was unsurprising that the simple linear models were unable to perform up to the task given the natural complexity/nonlinearity of the chemical reaction space, however the failure of the neural network-based models was, at least a little, unexpected. The immense power of various neural network architectures has been leveraged over recent years to tackle many chemical tasks akin to this one.[11,14,22,24,25] The unsurprising part is that these neural networks are much more complex than the double and triple layered perceptron models studied here, so the poorer performance of our simpler models could have been foreseen, however we leave for future exploration the possibility of leveraging a preexisting deep learning architecture for the yield prediction task at hand (*vide infra*). Of the three models that outperformed the one-hot baseline model, all three also outperformed the optimized models from Żurański et al.[2] From these three the lowest testing error comes from the *k*-nearest neighbor model with an RMSE of 12.3% yield, much of which Ahneman et al.[1] attributes to experimental measurement error.

Conclusions and recommendations

Summary

The prediction of reaction yields carries incredible significance in several realms of both academic and industrial synthetic chemistry. Machine learning methods and predictive modeling have been applied to field ranging from reaction and property prediction as well as rational catalyst design. The problem of reaction yield prediction being especially germane to industrial process chemistry, this project examined the 4140-sample high throughput screening set created by Doyle and coworkers. A plethora of models ranging from simple linear models to ensemble tree-based regression methods were evaluated with the aim of predicting Buchwald-Hartwig cross coupling yields in the presence of isoxazole additives. We were successful in modeling the reaction set with considerable accuracy and generalizability to out-of-sample molecules. The best performing model is an ℓ_1–distance-weighted 5-nearest neighbor regressor with 12.3% generalization error.

Further extensions

In addition to the questions raised in the results section we believe there are two major avenues for further extension of this project. Given the time and necessary computational resources and experience we would have done so ourselves, however we leave for further exploration the plethora of featurization methods and the effects of representation on model performance, as well as the expansion of the neural network class we very briefly examined in this project. The range of neural network architectures and their application to chemical problems has exploded in the past few years making this point incredibly germane.

Molecular featurization

In this study molecules were featurized using DFT, one of many available methods recently developed to turn molecules into machine-readable chemical meaning. In addition to the very high computational cost, DFT functionalization is sensitive to basis set and functional selection.[2] The dataset's molecular features were generated using the B3LYP/6-31G* level of theory, a general method used largely out of tradition.[1] Recent reports have demonstrated the significant shortcomings of this method.[26,27] To avoid the biases of this particular choice without recomputing every one of these features, the original DFT calculated features could be supplemented with several high (e.g. polar surface area, H-bond donor/acceptor ability, etc.) and low-level (i.e. molecular fingerprints) static representations using the molecular transformers in packages such as RDKit and DeepChem,[28] methodology which has recently been shown promising results when taking necessary precautions to avoid overweighting of a particular representaition.[13,14,29] Supplementing the original features with additional molecular descriptors could help to build models with greater predictive power.

Neural network architecture

In addition to sequence representations of the molecules in this project these inherently 3-dimensional objects also have graph representations.[11] In recent years the application of graph neural networks and message passing neural networks to organic chemistry has developed rapidly.[11,14,24] This dataset can easily be converted into graph representation (using the previously mentioned tools) and used to train various deep learning frameworks including message-passing and graph neural networks in some cases leveraging a preexisting architecture such as Chemprop[22] or MolNet[25] and in other cases building them anew.[14]

References

1. Ahneman, D. T., Estrada, J. G., Lin, S., Dreher, S. D. & Doyle, A. G. Predicting reaction performance in C–N cross-coupling using machine learning. *Science (80-.).* **360**, 186–190 (2018).
2. Żurański, A. M., Martinez Alvarado, J. I., Shields, B. J. & Doyle, A. G. Predicting Reaction Yields via Supervised Learning. *Acc. Chem. Res.* **54**, 1856–1865 (2021).
3. Gao, H. *et al.* Using Machine Learning To Predict Suitable Conditions for Organic Reactions. *ACS Cent. Sci.* **4**, 1465–1476 (2018).
4. Zahrt, A. F., Henle, J. J. & Denmark, S. E. Cautionary Guidelines for Machine Learning Studies with Combinatorial Datasets. *ACS Comb. Sci.* **22**, 586–591 (2020).
5. Lawrence, S. A. *Amines: synthesis, properties and applications*. (Cambridge University Press, 2004).
6. Kosugi, M., Kameyama, M. & Migita, T. Palladium-catalyzed aromatic amination of aryl bromides with N,N-diethylamino-tributyltin. *Chem. Lett.* **12**, 927–928 (1983).
7. Dorel, R., Grugel, C. P. & Haydl, A. M. The Buchwald–Hartwig Amination After 25 Years. *Angew. Chemie Int. Ed.* **58**, 17118–17129 (2019).
8. Ruiz-Castillo, P. & Buchwald, S. L. Applications of Palladium-Catalyzed C–N Cross-Coupling Reactions. *Chem. Rev.* **116**, 12564–12649 (2016).
9. Chuang, K. V. & Keiser, M. J. Comment on "Predicting reaction performance in C–N cross-coupling using machine learning". *Science (80-.).* **362**, eaat8603 (2018).
10. Lowe, D. M. Extraction of chemical structures and reactions from the literature. (2012).

doi:10.17863/CAM.16293.

11. Stokes, J. M. *et al.* A Deep Learning Approach to Antibiotic Discovery. *Cell* **180**, 688-702.e13 (2020).

12. Dehmer, M., Varmuza, K., Bonchev, D. & Emmert-Streib, F. *Statistical Modelling of Molecular Descriptors in QSAR/QSPR*. (Wiley, 2012).

13. Schwaller, P., Vaucher, A. C., Laino, T. & Reymond, J.-L. Prediction of chemical reaction yields using deep learning. *Mach. Learn. Sci. Technol.* **2**, 15016 (2021).

14. Pattanaik, L. & Coley, C. W. Molecular Representation: Going Long on Fingerprints. *Chem* **6**, 1204–1207 (2020).

15. Breiman, L. Random Forests. *Mach. Learn.* **45**, 5–32 (2001).

16. Friedman, J. H. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Stat.* **29**, 1189–1232 (2001).

17. Freund, Y. & Schapire, R. E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.* **55**, 119–139 (1997).

18. Hastie, T., Tibshirani, R. & Friedman, J. H. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. (Springer, 2009).

19. Murphy, K. P. *Machine Learning, a Probabilistic Perspective*. (MIT Press, 2012). doi:10.1080/09332480.2014.914768.

20. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in {P}ython. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).

21. Chen, T. & Guestrin, C. XGBoost. in *Proceedings of the 22nd {ACM} {SIGKDD} International Conference on Knowledge Discovery and Data Mining* (ACM, 2016). doi:10.1145/2939672.2939785.

22. Yang, K. *et al.* Analyzing Learned Molecular Representations for Property Prediction. *J. Chem. Inf. Model.* **59**, 3370–3388 (2019).

23. Hanaoka, K. Deep Neural Networks for Multicomponent Molecular Systems. *ACS Omega* **5**, 21042–21053 (2020).

24. Huang, K. *et al.* Therapeutics Data Commons: Machine Learning Datasets and Tasks for Drug Discovery and Development. (2021) doi:10.48550/ARXIV.2102.09548.

25. Kim, Y. *et al.* MolNet: A Chemically Intuitive Graph Neural Network for Prediction of Molecular Properties. (2022) doi:10.48550/ARXIV.2203.09456.

26. Morgante, P. & Peverati, R. The devil in the details: A tutorial review on some undervalued aspects of density functional theory calculations. *Int. J. Quantum Chem.* **120**, e26332 (2020).

27. Kruse, H., Goerigk, L. & Grimme, S. Why the Standard B3LYP/6-31G* Model Chemistry Should Not Be Used in DFT Calculations of Molecular Thermochemistry: Understanding and Correcting the Problem. *J. Org. Chem.* **77**, 10824–10834 (2012).

28. Ramsundar, B. *et al. Deep Learning for the Life Sciences*. (O'Reilly Media, 2019).

29. Sandfort, F., Strieth-Kalthoff, F., Kühnemund, M., Beecks, C. & Glorius, F. A Structure-Based Platform for Predicting Chemical Reactivity. *Chem* **6**, 1379–1390 (2020).

Abbreviations

Ad: adamantyl (–$C_{10}H_{15}$); API: active pharmaceutical ingredient; ^{t}Bu: *tert*-butyl (–$C(CH_3)_3$); Cy: cyclohexyl (–C_6H_{11}); DFT: density functional theory; HTS: high throughput screening; Me: methyl (–CH_3); Ph: phenyl (–C_6H_5); ^{i}Pr: iso-propyl (–$CH(CH_3)_2$); QSAR: quantitative structure–activity relationship; QSPR: quantitative structure–property relationship; TDC: Therapeutics Data Commons