Predicting the Yields of Suzuki Coupling Reactions Using Machine Learning

Stephen Marriott

Brown University Department of Chemistry and Data Science Institute

github.com/Stephen-Marriott/DATA1030_Suzuki

Introduction

The Suzuki-Miyaura cross coupling reaction is a carbon-carbon bond forming reaction widely regarded as being one of the most significant transformations of its type available to modern-day organic chemists.[1] To further illustrate the reaction's importance is the fact that it constitutes part of the work awarded the Nobel Prize in chemistry in 2010.[2] The reaction introduces a carbon-carbon bond between two organic fragments, which is highly exploitable in industry for the synthesis of various important products such as pharmaceuticals.[3]

Despite the robustness of this system, no reaction in chemistry is guaranteed to work with all possible combinations of input variables. As such, the success of a Suzuki coupling with specific inputs, best captured by the reaction yield ($\frac{obtained\ product\ (mol)}{expected\ product\ from\ complete\ conversion\ (mol)}$), can't usually be known until after the reaction has been run. Industrially, this results in significant trial and error to optimise reaction conditions, which is both costly and time consuming. *In silico* reaction

condition screening could use machine learning to train models that predict the outcome of a variable combination, which would provide some insight into which reaction conditions are most worthwhile pursuing without attempting the reaction chemically.

In 2018, Perera et al. published a set of over 5000 unique Suzuki coupling reactions collected by flow high-throughput screening.[4] Tabularised by github user “leojklarner”, the data contains the reaction yields (our chosen target variable) for most unique combinations of the 5 different input variables (*Figure 1*).[5] These features are the two substrates (a quinoline and an indazole) ‘*reactant_1_smiles*’ and ‘*reactant_2_smiles*’, the ligand ‘*ligand_smiles*’, the base ‘*reagent_1_smiles*’ and the solvent medium ‘*solvent_1_smiles*’. All are categorical, consisting of SMILES string representations of the chemical categories. Other features included in the dataframe are the catalyst used ‘*catalyst_smiles’* and the unique representation of the ‘*rxn*’.

Herein we investigate various machine learning methods for use in predicting the reaction yields within Perera’s dataset.

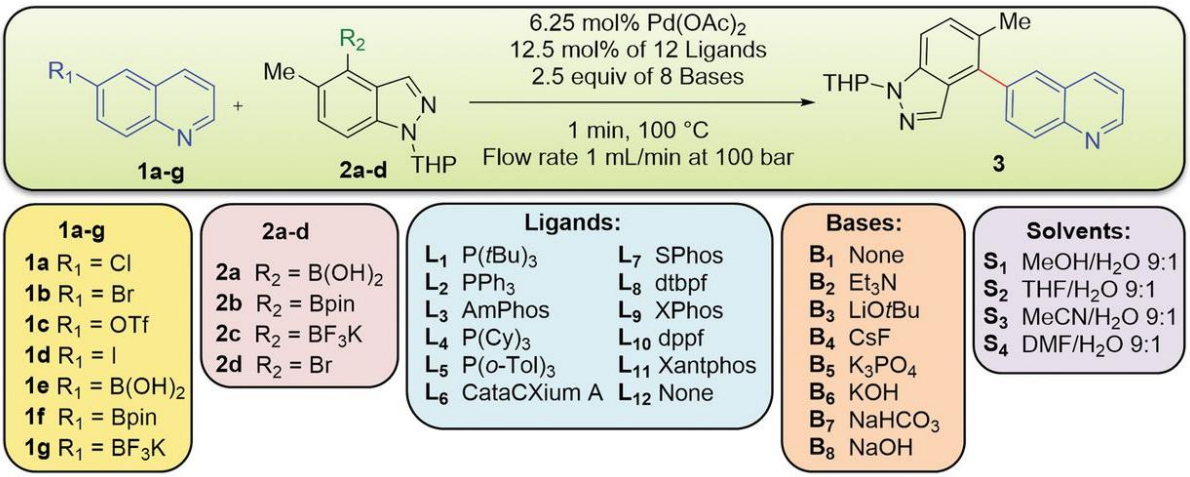

Figure 1. A depiction of the Suzuki-Miyaura cross coupling investigated by Perera et al., and the various categories tested for each of the 5 features.

EDA

We first examined the distributions of the target and the features in the dataset. Value counts for the number of categories for the feature '*rxn*' returned 5760, which is the number of rows in the table, suggesting that this feature contains no useful additional categorical information. A bar plot for the feature '*catalyst_smiles*' shows that every row has the same category, which also suggests this feature has no useful information (*Figure 2*). As such, the decision was made to drop these two features. A histogram of the target variable (*Figure 3*) shows a roughly positively-tailed distribution of yields with mean of 0.401 and standard deviation 0.281, with large spikes around 0.17 and for 0-yielding reactions, indicating that there could be one or more reactants or conditions that are responsible for killing the reaction.

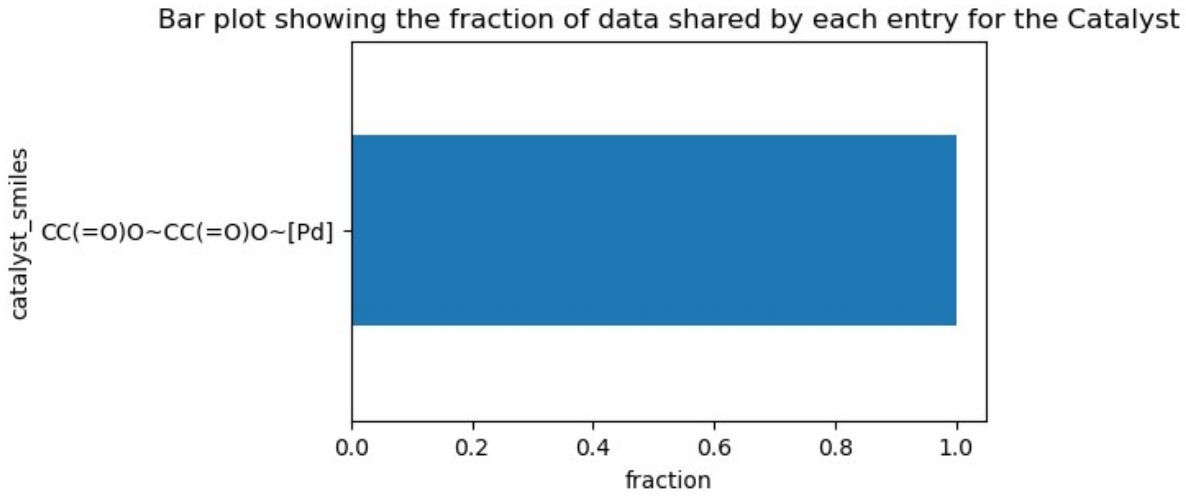

Figure 2. A barplot showing the distribution of data in the categorical feature '*catalyst_smiles*'

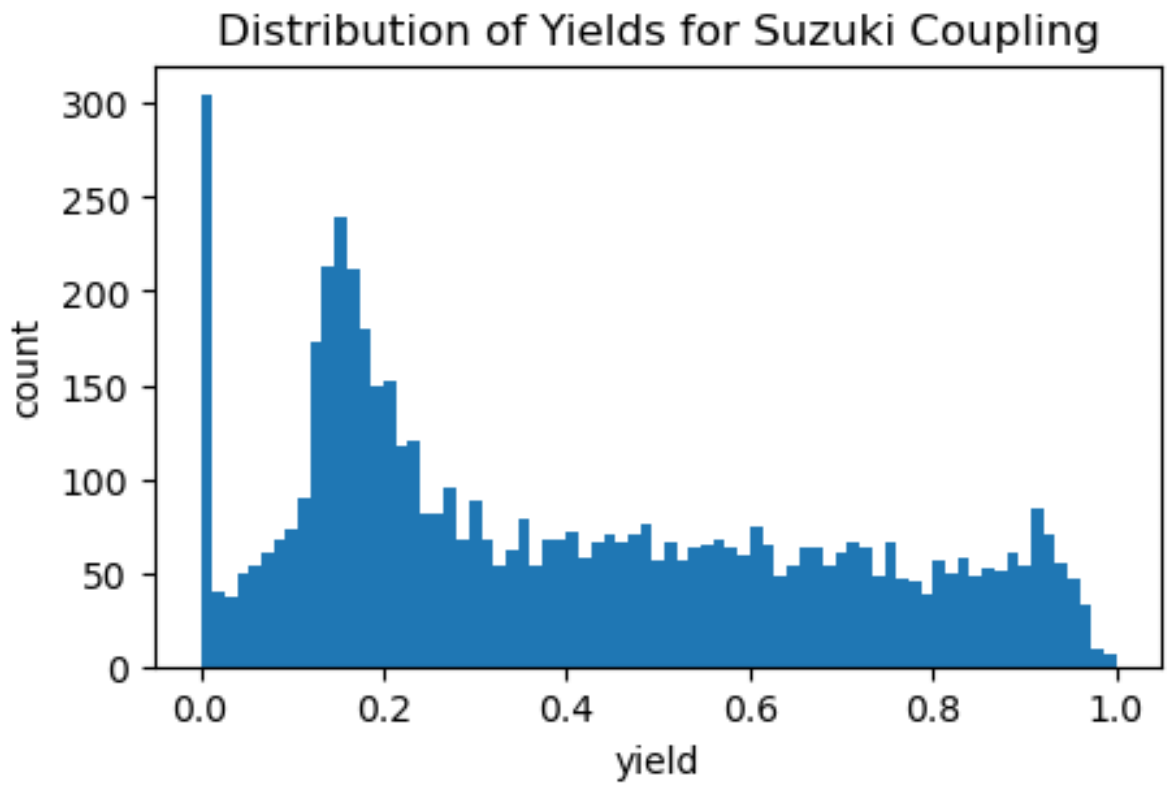

Figure 3. A histogram showing the distribution of the reaction yields from the Suzuki-Miyaura dataset.

Looking more closely at the approved features we can see from the barplots that all the categorical features are evenly distributed amongst each category (*Figures 4-8*). The least even of these features, Reactant 1, only has a minimum class percentage of 6.7% for its minority classes, which is not significant enough of a class imbalance to warrant the use of stratified sampling.

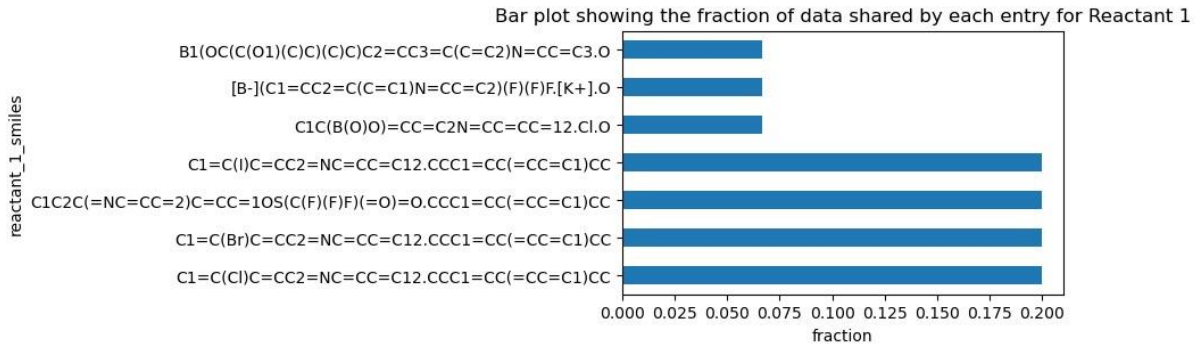

Figure 4. A barplot showing the distribution of data in the categorical feature '*reactant_1_smiles*'

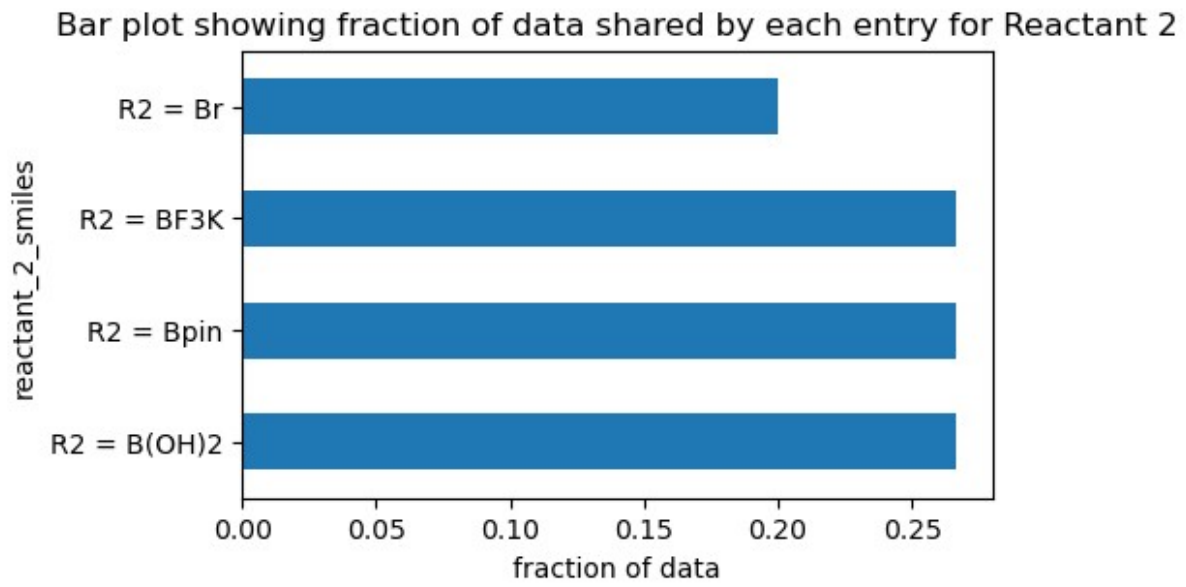

Figure 5. A barplot showing the distribution of data in the categorical feature '*reactant_2_smiles*'

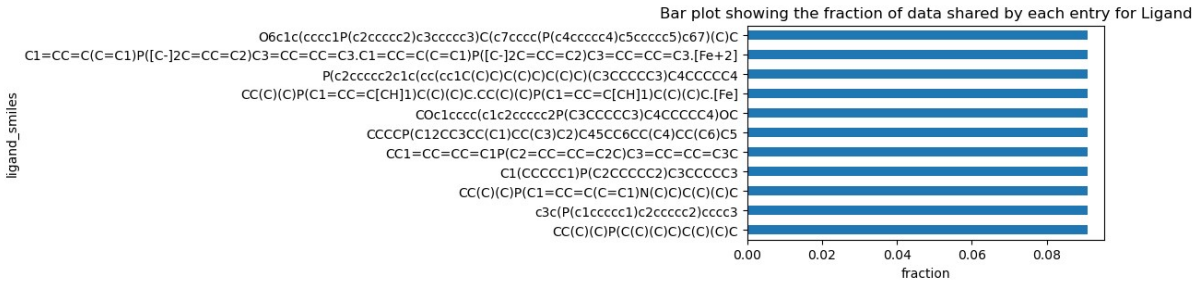

Figure 6. A barplot showing the distribution of data in the categorical feature '*ligand_smiles*'

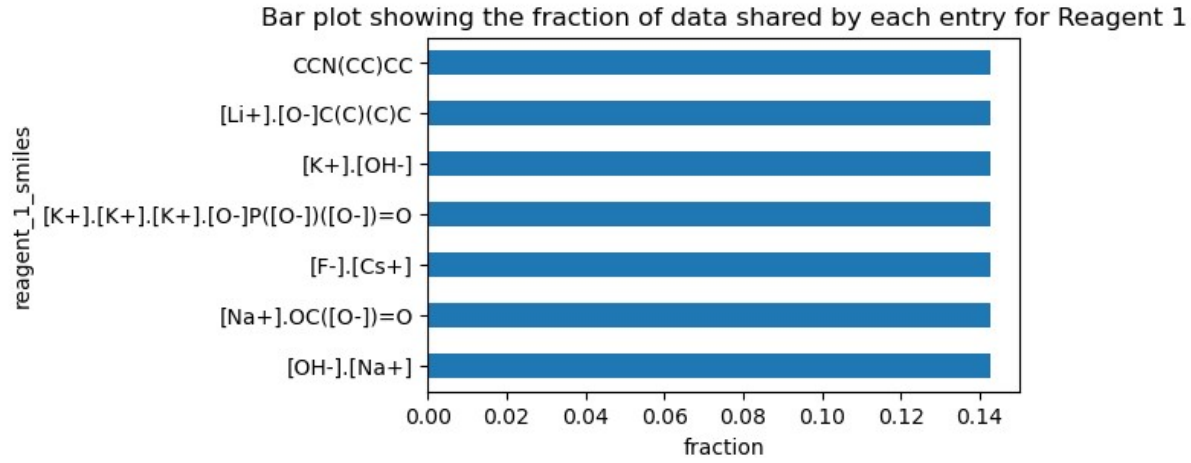

Figure 7. A barplot showing the distribution of data in the categorical feature '*reagent_1_smiles*'

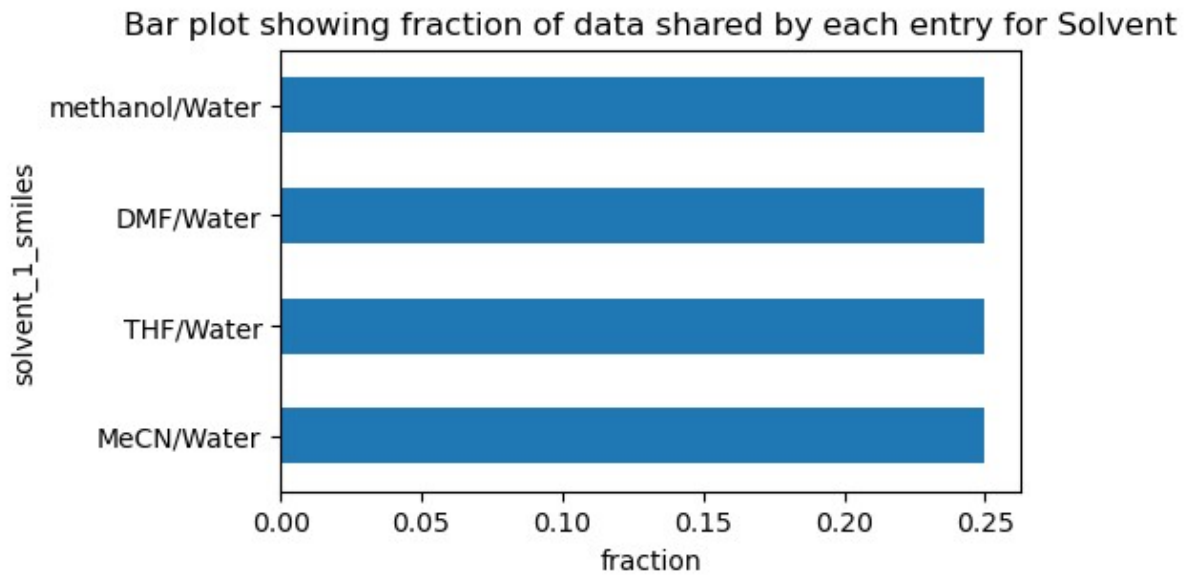

Figure 8. A barplot showing the distribution of data in the categorical feature '*solvent_smiles*'

It was also found that 19.8% of the rows in the table contained missing values, concentrated in the features *'ligand_smiles'* and '*reagent_1_smiles'.* Luckily, as these are both categorical features, the SimpleImputer could be used to treat these values as a separate category. Furthermore, preprocessing of these features was simply done with the OneHotEncoder, resulting in a final feature count of 35 starting from an original 7. A Pearson correlation matrix was calculated for these 35 features (following basic splitting) (*Figure 9*). It shows very little cross-feature correlation, besides negative correlation between categories belonging to the same feature (which is expected), and slight positive correlation between some categories in the features '*reactant_1_smiles'* and '*reactant_2_smiles'* which is a result of the necessity of boronates being paired with a halide/tosylate. Therefore, there was no need to drop any further features from correlation.

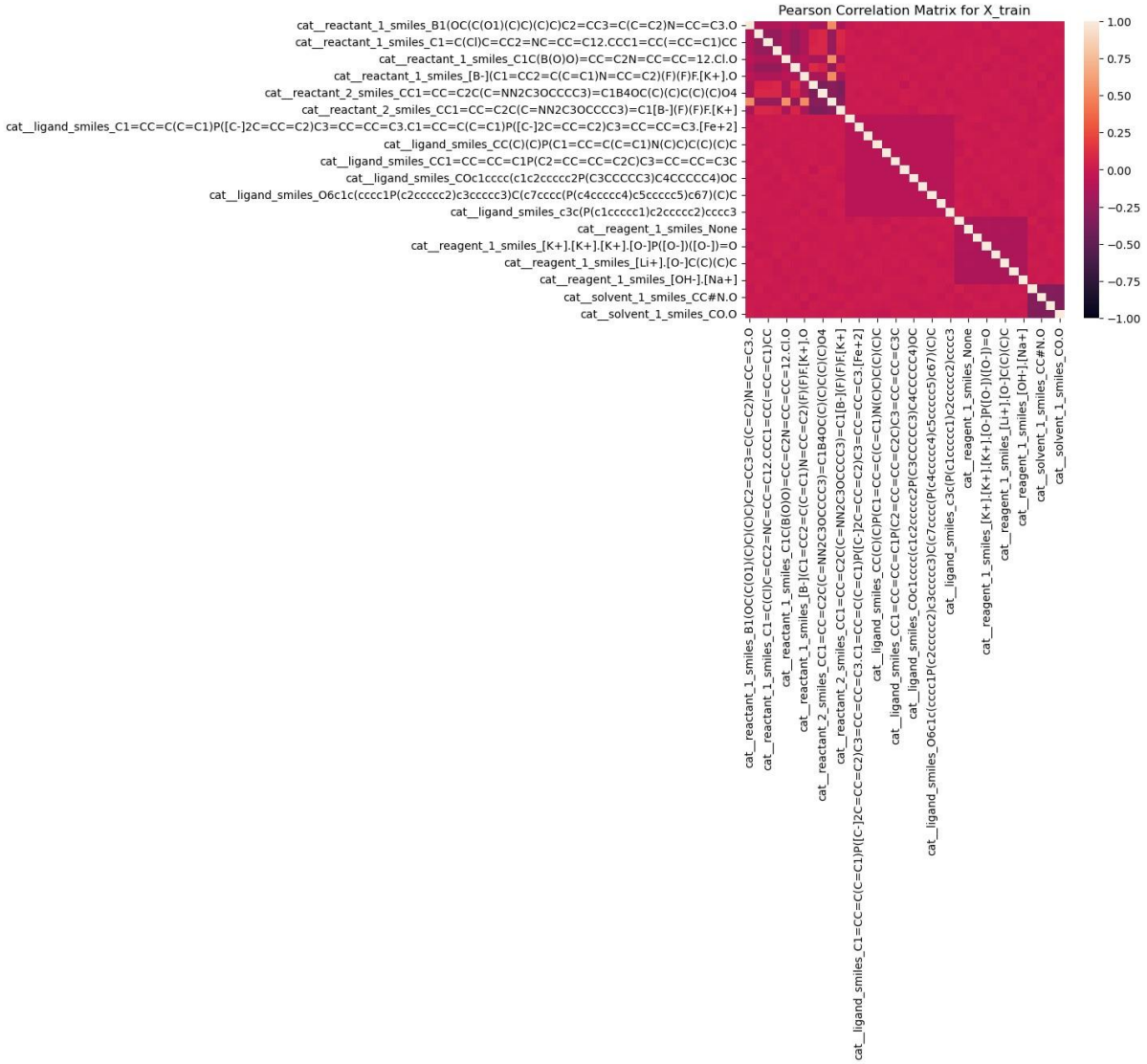

Figure 9. A Pearson correlation matrix of the preprocessed features from the train split of the data.

Methods

Normal train_test_split was implemented, reserving 20% of the data for testing. Shuffling was ensured due to the dataframe being organised in order of category within each feature. As the dataset was quite small, it was decided to use normal KFold splitting (4 folds) to maximise the utilisation of the data. As discussed above, stratified splitting was decided against, as was group splitting. Again, shuffling was ensured.

As aforementioned, the preprocessing of the split data involved imputing the missing values and then using OneHotEncoder to convert the categorical string features into numerical features.

The splitting was done within a function designed to loop over 10 different random states (to minimise the effect of uncertainty). The split data, along with an initialised machine learning model, was then fed to a GridSearchCV algorithm that permutes through all possible parameter combinations from a given parameter grid input and finds the optimal hyperparameters. The best model from this cross-validation search was used to calculate a training score, which was averaged over the 10 iterations and compared against a baseline and the performance of other models.

The metric in question was the mean squared error, due to the desire to penalise predictive errors more. This is because the purpose of the model is to determine whether a set of reaction conditions is worth pursuing or not to save money and time, and predictive errors will undermine this aim. Furthermore, as the target variable has no units, there is no need to use the root mean squared error to maintain the same units.

We tested 6 machine learning models using the above pipeline: lasso, ridge and elastic net (linear models) and random forest, support vector machine and k-neighbours (non-linear models). Before running the GridSearchCV on these models, the train and validation scores were calculated for hyperparameters with no clear bounds to find the best range to test in the cross validation (*Figures 10-17*).

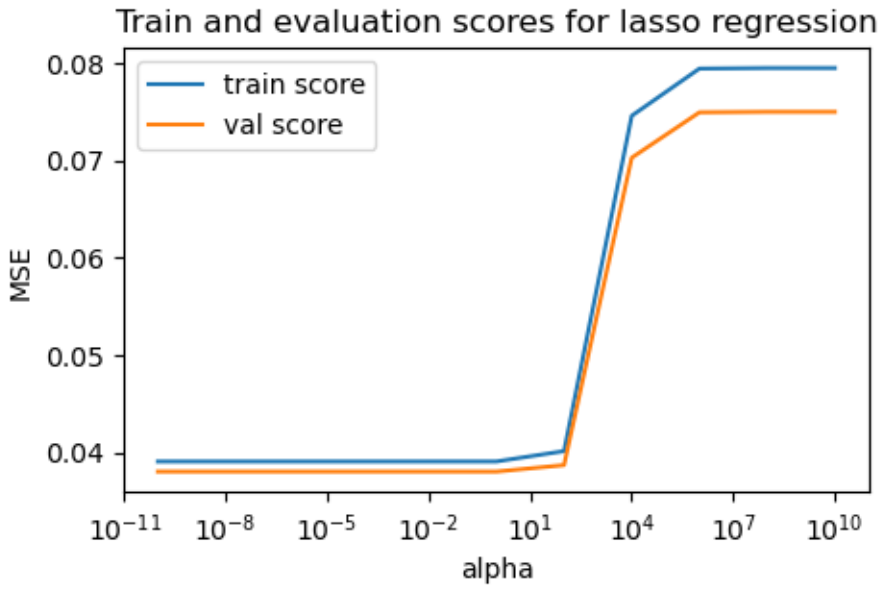

Figure 10. A plot showing the train and validation curves for the alpha hyperparameter of lasso regression.

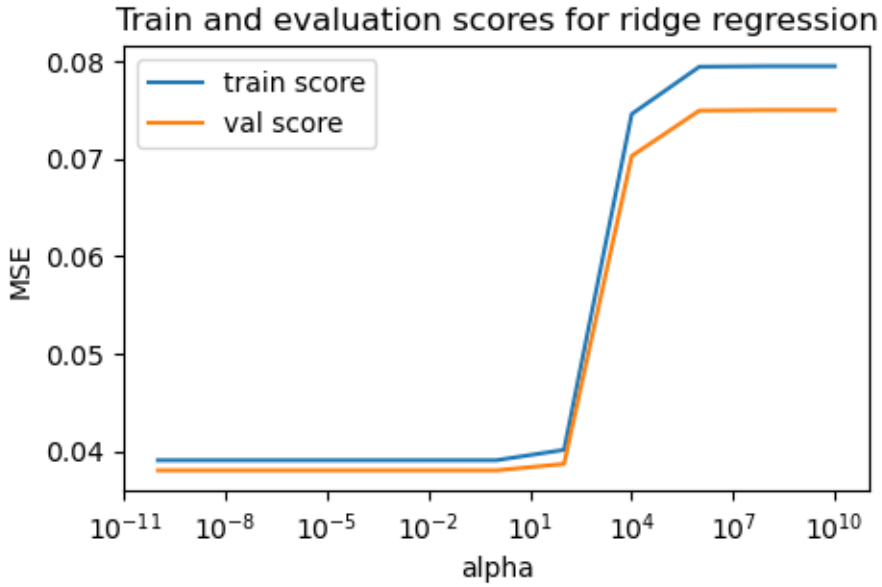

Figure 11. A plot showing the train and validation curves for the alpha hyperparameter of ridge regression.

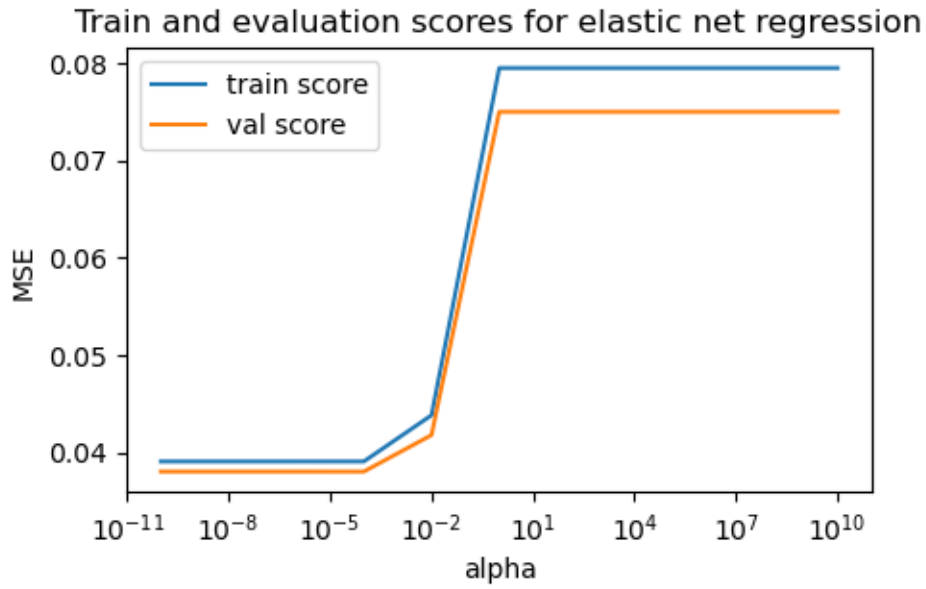

Figure 12. A plot showing the train and validation curves for the alpha hyperparameter of elastic net regression.

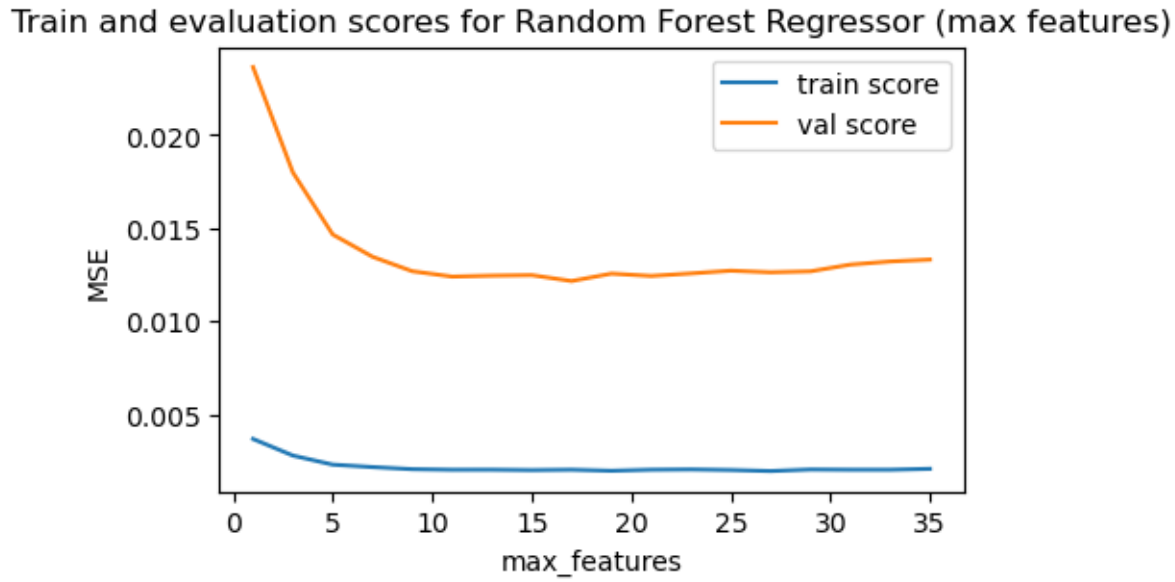

Figure 13. A plot showing the train and validation curves for the max-features hyperparameter of random forest regression.

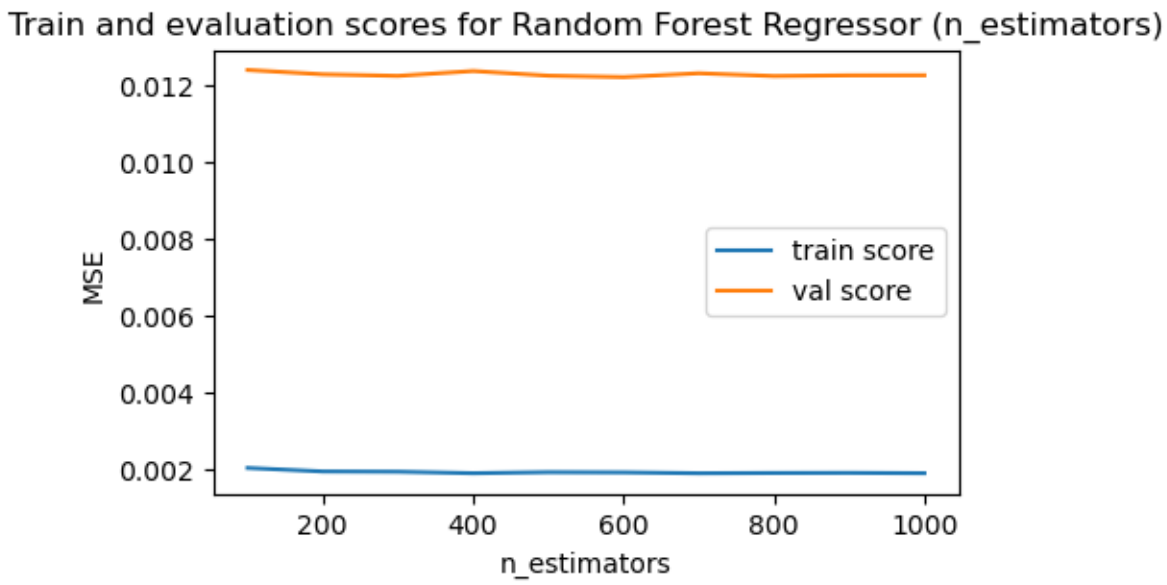

Figure 14. A plot showing the train and validation curves for the n_estimators hyperparameter of random forest regression.

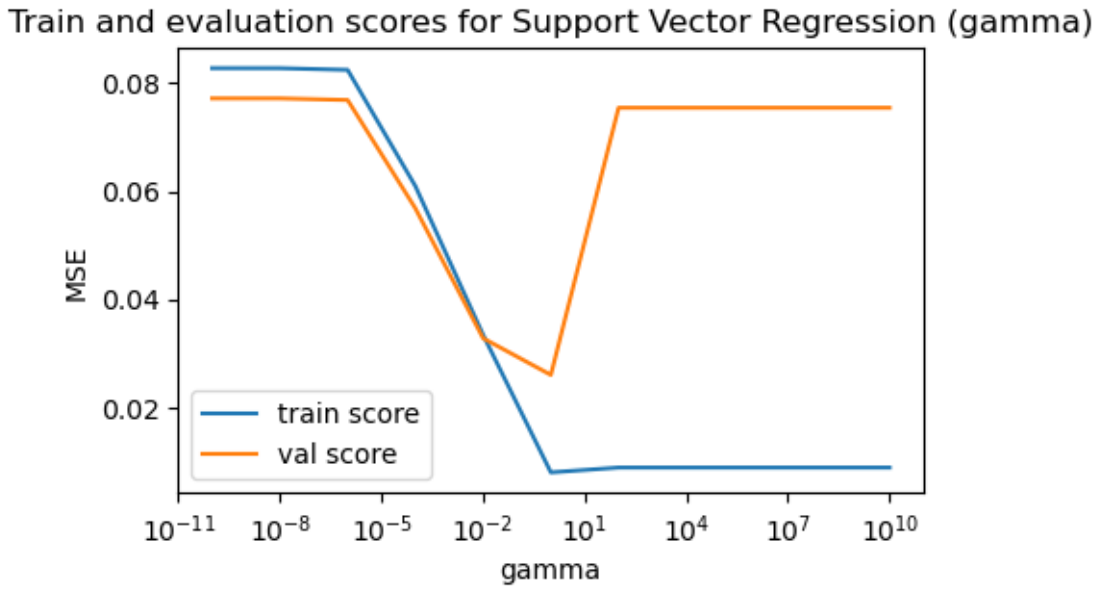

Figure 15. A plot showing the train and validation curves for the gamma hyperparameter of support vector regression.

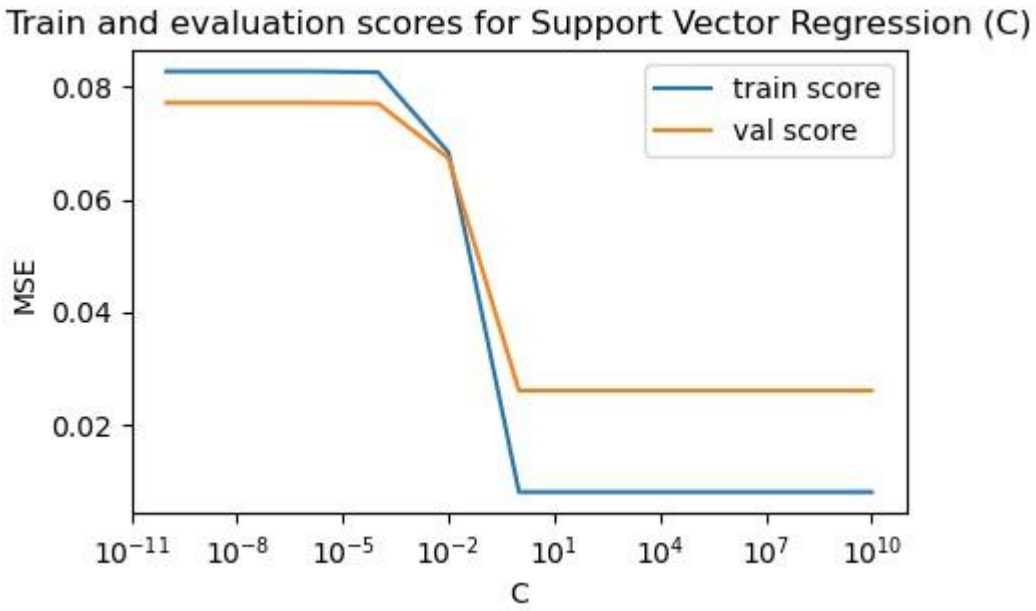

Figure 16. A plot showing the train and validation curves for the C hyperparameter of support vector regression.

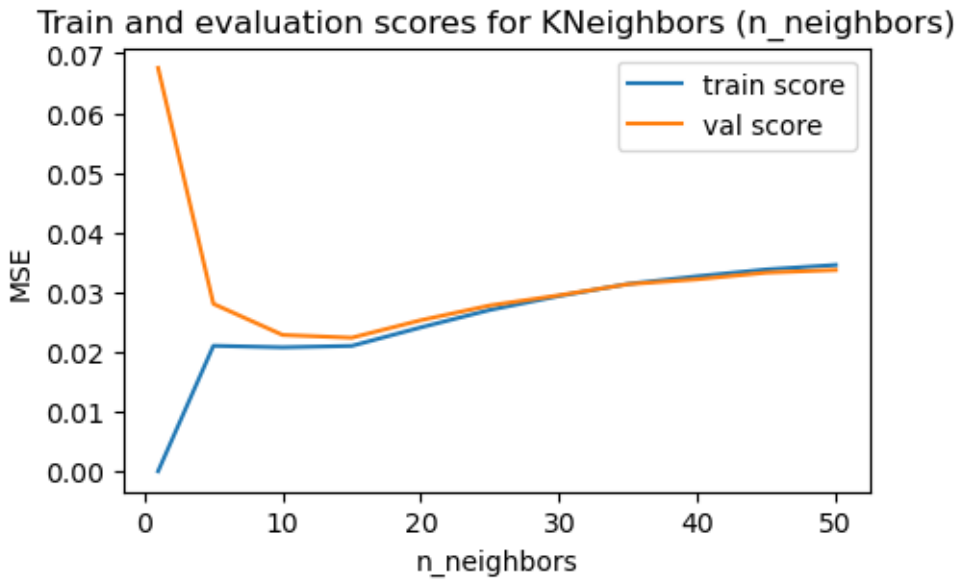

Figure 17. A plot showing the train and validation curves for the n_neighbors hyperparameter of KNeighbors regression.

Following this initial hyperparameter range screening, hyperparameter grids were built for each model and the pipeline was run, finding the optimal hyperparameter combinations (*Table 1*) and retrieving the test scores.

Machine Learning Model	Hyperparameters tested
Lasso	alpha = 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3, 1e4, 1e5, 1e6, 1e7
Ridge	alpha = 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3, 1e4, 1e5, 1e6, 1e7
Elastic Net	alpha = 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1 L1-ratio = 0, 0.2, 0.4, 0.6, 0.8, 1
Random Forest	Max_depth = 3, 5, 10, None Max_features = 1, 5, 10, 15, 20, 25, None
Support Vector Machine	C = 1e-2, 1e-1, 1e0, 1e1, 1e2 gamma = 1e-2, 1e-1, 1e0, 1e1, 1e2
K Neighbors	Metric = 'euclidean', 'manhattan', 'minkowski' n_neighbors = 5, 7, 9, 11, 13, 15, 17, 19, 21 Weights = 'uniform', 'distance'

Table 1. A showing the hyperparameters and values used in the GridSearchCV for each of the tested machine learning models. Highlighted in red are the optimal hyperparameters found.

Results

A baseline mean squared error score for the model was calculated using a dummy regressor that predicts the mean for every value. The mean test scores and standard deviations calculated for each of the optimised models were plotted against the baseline (*Figure 18*). All models performed better than the baseline (lower error), with the three non-linear models being better than the linear models. Interestingly, the Ridge regression performed just as well as the Elastic Net, likely because the optimised L1-ratio was 0. The best performing model was the Random Forest

Regressor. The yields were predicted using the optimised Random Forest and then plotted against the true values (*Figure 19*). The plot shows that the model performs well at very high and also close to 0 yields, with larger variance occurring in intermediate yield ranges. This is likely due to the larger amount of data available at higher and lower yields (so better predictions can be made), as seen in *Figure 3*. One interesting feature of the plot is the somewhat poor performance at predicting 0-yielding reactions.

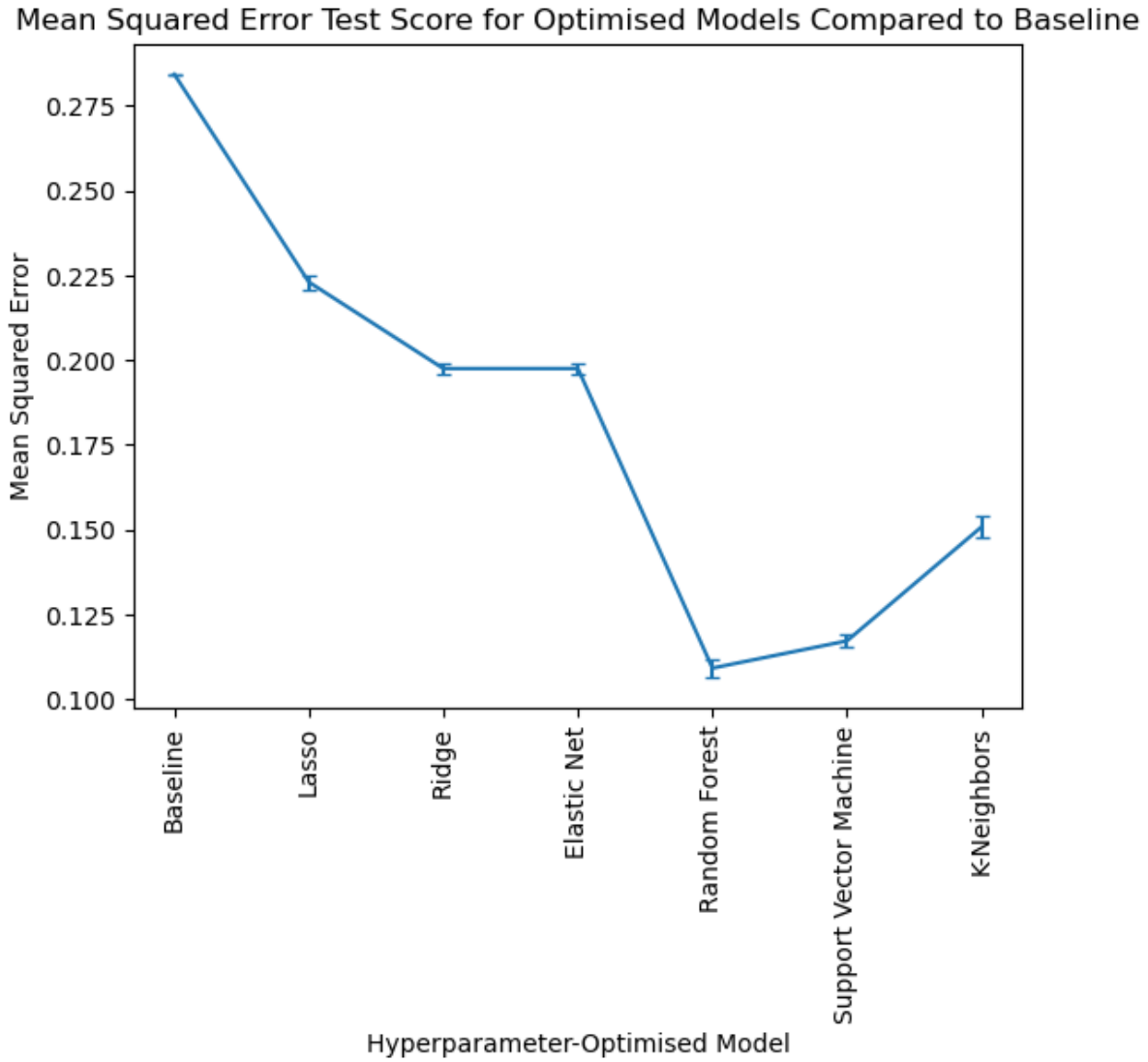

Figure 18. A plot showing the mean squared error test score means and standard deviations for the optimised machine learning models compared to the baseline.

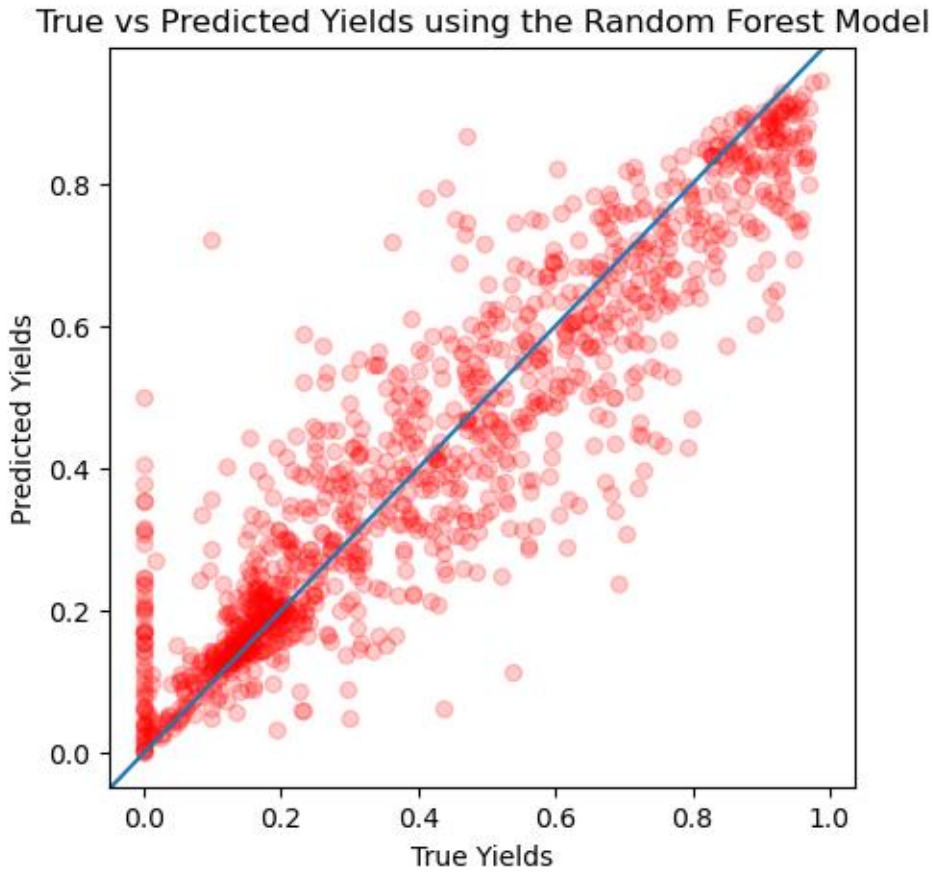

Figure 19. A plot showing the predicted target values against the true target vales for the optimised random forest model. Opaquer regions indicate greater point density. The blue line indicates the perfect predictor.

The global feature importance was calculated using Mean Decrease of Impurity (MDI) (*Figure 20*), permutation importance (*Figure 21*) and Shapley Additive Explanations (SHAP) (*Figure 22*), and the top 10 features were plotted. For each of these measures of importance, the same 4 features appeared as the most important: reactant_1_smiles_C1=C(Cl)C=CC2=NC=CC=C12.CCC1=CC(=CC=C1)CC, reactant_1_smiles_C1=C(I)C=CC2=NC=CC=C12.CCC1=CC(=CC=C1)CC, reactant_2_smiles_CC1=CC=C2C(C=NN2C3OCCCC3)=C1[B-](F)(F)F.[K+], and ligand_smiles_O6c1c(cccc1P(c2ccccc2)c3ccccc3)C(c7cccc(P(c4ccccc4)c5ccccc5)c67)(C)C. Some of the least important features include

reactant_1_smiles_[B-](C1=CC2=C(C=C1)N=CC=C2)(F)(F)F.[K+].O and reagent_1_smiles_[K+][OH-].

Figure 20. A barplot showing the top 10 global feature importances calculated by MDI.

Figure 21. A barplot showing the top 10 global feature importances calculated by permutation importance.

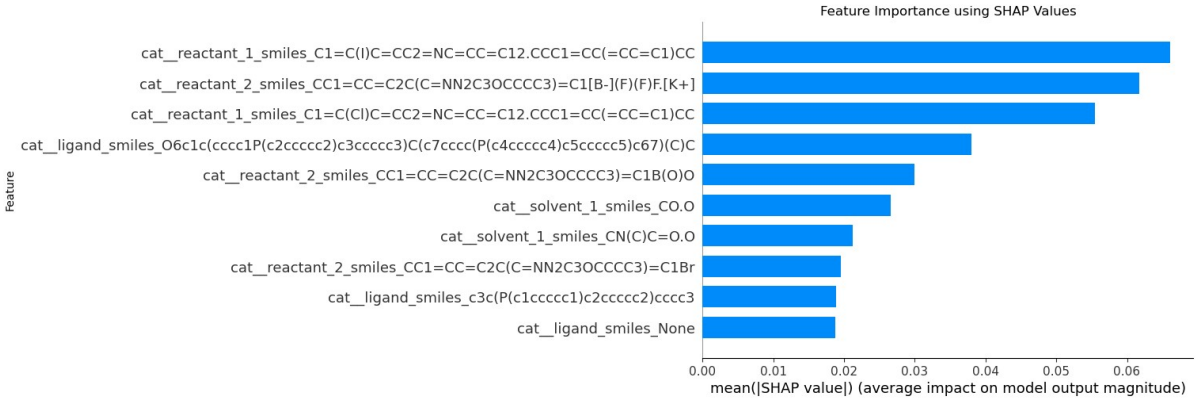

Figure 22. A barplot showing the top 10 global feature importances calculated by SHAP values.

Local feature importance was also calculated using SHAP values, and the values displayed on force plots (Figures 23-26). From these plots, we can see that of the top 4 features, all but reactant_1_smiles_C1=C(I)C=CC2=NC=CC=C12.CCC1=CC(=CC=C1)CC contribute negatively to the overall reaction yield, suggesting that their presence in the reaction is particularly bad.

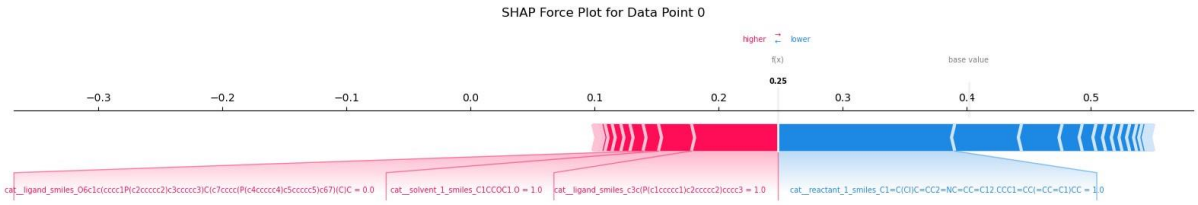

Figure 23. A force plot showing the SHAP local feature importances for data point 0.

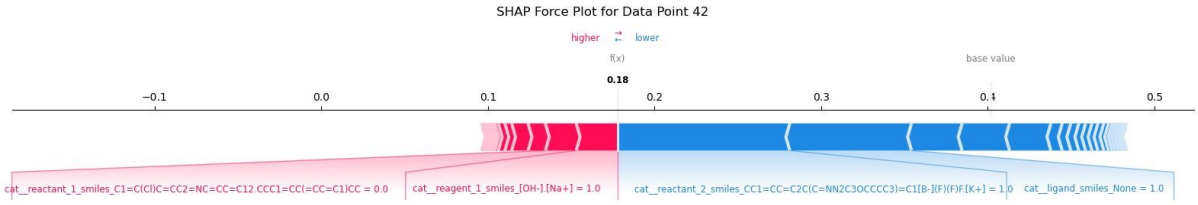

Figure 24. A force plot showing the SHAP local feature importances for data point 42.

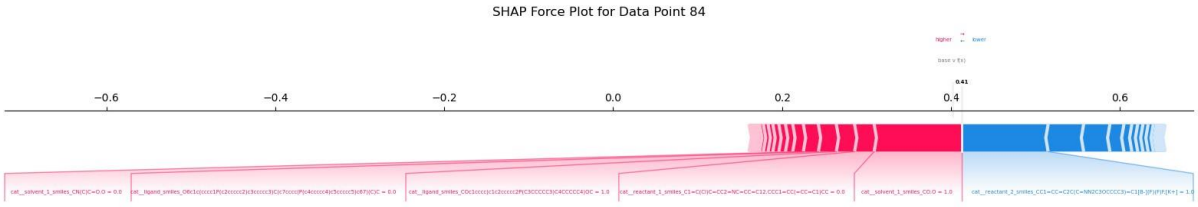

Figure 25. A force plot showing the SHAP local feature importances for data point 84.

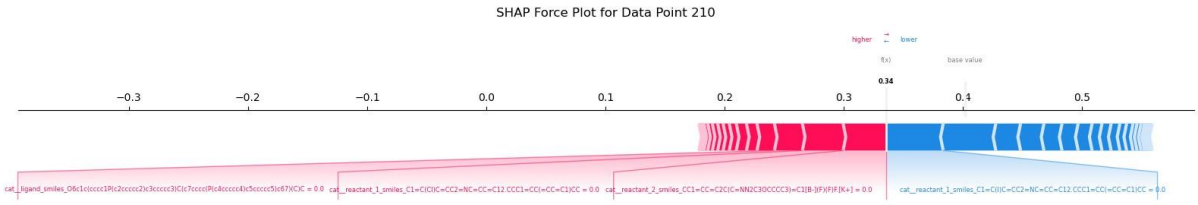

Figure 26. A force plot showing the SHAP local feature importances for data point 210.

Outlook

The weakest component of this project is the limited nature of the dataset; the substrates tested are very similar in structure, only varying by minor functional group changes. For the models to be more useful in general applications, it would be critical to incorporate further datasets with more varied substrate structures. With information on a greater range of substrates, it would be possible to featurise the SMILES strings to derive more general feature descriptors, such as the count of different functional groups. Regarding methods, given more time I certainly could have attempted to use more powerful machine learning models, such as XGBoost and ensemble methods.

References

1) Barder, T. E., Walker, S. D., Martinelli, J. R., Buchwald, S. L., *J. Am. Chem. Soc.* **2005**, 127, 13, 4685–4696
2) The Nobel Prize in Chemistry 2010. NobelPrize.org. Nobel Prize Outreach AB 2023. Sat. 9 Dec 2023, https://www.nobelprize.org/prizes/chemistry/2010/summary/
3) Miyaura, N.; Yamada, K.; Suzuki, A. Tetrahedron Lett. 1979, 36, 3437-3440

4) Perera et al., *Science* 359, 429–434 (**2018**), doi.org/10.1126/science.aap9112
5) github.com/leojklarner/gauche/blob/main/data/reactions/suzuki_miyaura_data.csv