Project Report
ChBE 6745
Spring 2020

Department of Chemical and Biomolecular Engineering
Georgia Institute of Technology

Developing a Predictive Capability of Mechanical Stability in Additive-Containing PET Fibers

Group 08: Aaron Liu
Brian Khau
Max Bukhovko
Nicole (Yuge) Hu
Rahul Venkatesh

April 18, 2020

Contents

1 Introduction

Poly(ethylene terephthalate) (PET) and other polyesters are ubiquitous in the commercial thermoplastics industry. These materials are used for a variety of applications, ranging from thermoformed containers for packaging to woven or extruded fibers for clothing. These plastics are often employed due to their superior mechanical properties. For these materials, the incorporation of thermo-oxidative stabilizers is an important design consideration in applications where mechanical properties (tenacity, tensile strength, elongation, etc.) degrade over time due to heat exposure. While commercially available additives are generally effective in mitigating thermal degradation, maximizing this mitigation for an application-specific time and temperature exposure via optimal additive choice can be time and resource intensive. Thus, a contemporary problem is the discovery of small-molecule additives that can adequately improve the mechanical property persistence of polyester fibers through thermal exposure.

The main objective of this project is to construct a framework to predict the performance of small molecules as thermal stabilizing additives in PET. Here, we use a dataset that reports a set of candidate additives and the resulting mechanical property (tenacity) persistence of PET after heat exposure. Tenacity is a measure of a fiber's ultimate breaking strength; the higher the tenacity, the more resistant the fiber is against breaking. This dataset also includes a large design space (1,862 features) of molecular descriptors, which for the purposes of this project are considered already provided and not directly calculated as part of this project.

Details of dataset are as follows:

- The data in the attached file `PETadditives_database_and_descriptors.csv` was previously mined from patent US 3,491,057.
- Each row represents an additive-containing PET fiber sample.
- An addition method (a categorical variable denoted by "A," "B," or "C") and the amount (in grams) of additive incorporated were specified for each observation.
- Each sample was subjected to both a "Wet Heat Treatment" (placed in a closed vessel of water and heated at 120 °C for 3, 5, and 15 days) and a "Dry Heat Treatment" (secured in a dry environment in air at 230 °C for 1, 2, and 3 hours). The tenacity of the sample was measured before and after the prescribed exposure time and reported as the "tenacity persistence," for a given column:

$$y = \frac{\text{tenacity after treatment}}{\text{tenacity before treatment}}(100\%)$$

- The features in the dataset (x_1, x_2, x_3, etc.) are molecular descriptors derived from the structure of the additive. These include descriptors such as molecular weight and branching, and were calculated in previous work using alvaDesc 1.0.12. The descriptors here were provided as part of the dataset. For a complete list of the molecular descriptors, please refer to the link.

Our final dataset for analysis in this project consists of 42 observations with 1,862 molecular descriptors. Since the independent variable in this study is the chemical identities of the additives, these molecular descriptors were calculated as a method to generate a feature space. While the mechanics of this calculation is beyond the scope of this project, the key takeaway is that molecular descriptors help to assign various structural features to different molecules and allow us to attempt to build a predictive model, or a **quantitative structure-activity relationship (QSAR)**. In other words, QSARs are regression or classification models that attempt to draw correlations between molecular structures and a desired property.

Main Objectives and Goals:

- To evaluate a baseline QSAR model for tenacity persistence prediction using multilinear regression approaches, such that the performance of subsequent models can be compared.
- Implement at least one form of dimensionality reduction to eliminate features that are redundant or have little information.
- Evaluate and compare improved QSAR models generated using L_1-regularization (LASSO),L_2-regularization Kernel Ridge Regularization (KRR) ,latent variable projection techniques (principal component analysis (PCA) or partial least squares regression (PLS)). This includes cross-validation in addition to hyperparameter optimization.
- Perform final validation of our models using a small subset of our data held-out in the beginning of the workflow.

If our QSAR model demonstrates predictive power, analogous model frameworks hypothetically could be used to screen a repository of new potential small-molecule additives and predict their performance, to be validated against laboratory experiments.

2 Methodology

2.1 Description of Model Validation Strategy

The raw dataset containing 63 experiments from the patent was imported using the pandas library and was filtered down to 42 samples that were processed with addition method A and 0.25 g of the candidate added, presenting the largest subset of data with consistent processing conditions. These 42 observations were subsequently used throughout this analysis. The tenacity persistence values considered for this project were derived from the column labeled "Dry Heat Treatment, 1 hour." The preprocessed dataset was split into training and validation datasets using a hold out percentage of 20%. Model development was then performed solely on the training dataset, in order to ensure that the validation set is not contaminated. Prior to model development, the training dataset was scaled using the standard scaler. After final model development, the validation set was scaled using the same parameters as the training dataset. For this project, five preliminary QSAR regression models have been fitted on the tenacity persistence dataset through hyperparameter tuning and appropriate feature selection.

2.2 Baseline Models (OLS, PCR)

First, we constructed a naive ordinary least squares (OLS) regression model using all available 1,862 features. To streamline the model further, we subsequently implemented a principal component analysis (PCA) on the feature space. Dimensionality reduction of the dataset was performed by retaining a number of principal components that cumulatively explain 90% of the variance in the dataset. Finally, we performed a principal component regression (PCR), where we implemented a linear regression model developed using this subset of principal components.

2.3 Improved Models (LASSO, PLSR, KRR)

Due to the high dimensionality of our feature space, we explored other methods of performing feature selection and dimensionality reduction, such as L_1-regularization (LASSO), Kernel Ridge Regularization(KRR) and Partial Least Squares Regression (PLSR). Additionally, we developed an artificial neural network (ANN) as an example of a highly-flexible and non-linear regressor and its results are present in Appendix B.

2.4 Cross-Validation

Since the dataset contained only 42 total observations, cross-validation via leave-one-out (LOO) was used to quantify error for the training and testing datasets. These models were then directly compared by examining their performance on three components: the training data, testing data, and the validation data. Hyperparameters were optimized by minimizing the root-mean-squared error of the predicted values vs the actual dataset. We have reported the RMSE and the R^2 values on the training and validation datasets.

2.5 Data Analytics Workflow

Figure 1 below depicts the workflow adopted for this project. This is composed of four main sections:

1. Data preprocessing.
2. Creating a validation dataset.
3. Pipeline development, where we define and optimize model parameters.
4. Applying our retrained model on our validation data.

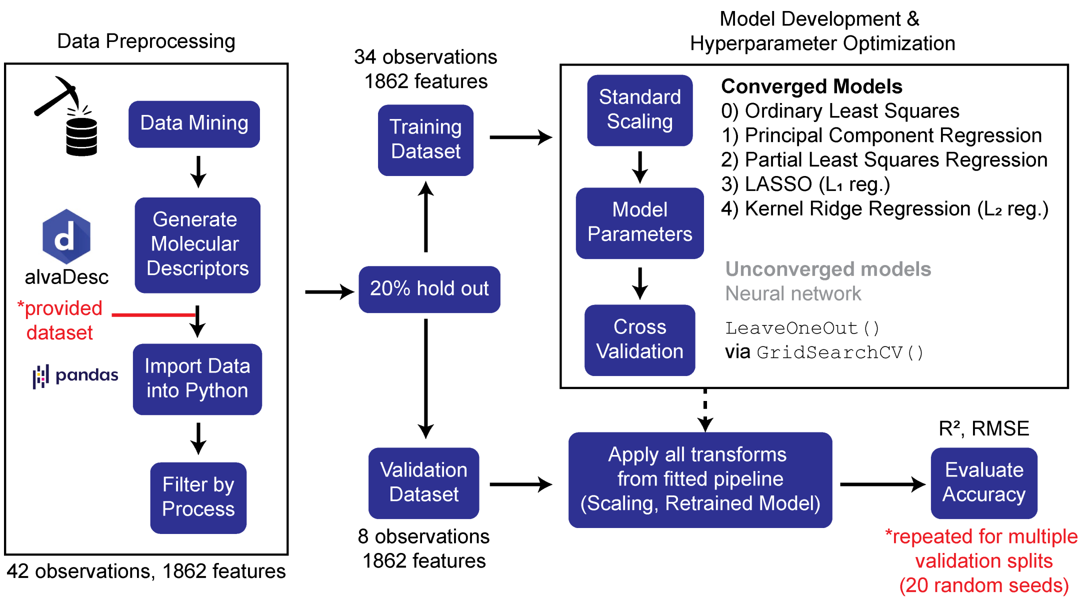


Figure 1: Workflow adopted for this project.

2.6 Brief Discussion of Software Packages or Algorithms Used

We used the `Pipeline()` class to wrap together all our transforms and estimators, including scaling. In addition, we imported `scikit-learn` functions for implementations whenever possible (such as `PCA()`). The pipeline description for each regression model used is shown below:

- Naive model: Ordinary Least Squares (OLS).
  - `pipeline = StandardScaler(), LinearRegression()`
- Streamlined Baseline Model: Principal Component Regression (PCR).
  - `pipeline = StandardScaler(), PCA(), LinearRegression()`
- Improved Model 1: Partial-Least-Squares regression (PLSR).
  - `Pipeline = StandardScaler(), PLSRegression()`
- Improved Model 2: L_1-regularized least-squares (LASSO).
  - `pipeline = StandardScaler(), Lasso()`
- Improved Model 3: L_2-regularized least-squares (Kernel Ridge Regression, KRR).
  - `Pipeline = StandardScaler(), KRR()`

We also implemented `LeaveOneOut()` implicitly through `GridSearchCV()` to aid in hyperparameter tuning. This enables us to select appropriate hyperparameters, such as the number of principal components or regularization strength.

3 Results and Discussion

3.1 Sensitivity Analysis Summary

Considering the low number of data points in our dataset, we found that the results are strongly influenced by the initial test-train hold out. We performed a sensitivity analysis for all of the models by running 20 different training/validation sets with different random seeds (see Table 1). This step is necessary because our small dataset is highly sensitive to how the validation data is split. An example of the visualization of these models are provided using `np.random.seed(40)` (Appendix A).

Table 1: Sensitivity analysis of all algorithms with 20 random numbers.

Model	Training R^2	Validation R^2	Training RMSE	Validation RMSE
OLS	1.0000 ± 0.0000	-0.8255 ± 1.1575	0.0000 ± 0.0000	3.1929 ±0.7589
PCR	0.4487 ± 0.2624	-0.4981 ± 0.8818	1.8379 ± 0.4018	2.8165 ± 0.6283
PLSR	0.8917 ± 0.1232	-0.5364 ± 0.8038	0.6169 ± 0.5341	2.9840 ± 0.6932
LASSO	0.8526 ± 0.2999	-1.3356 ± 3.5812	0.5119 ± 0.7780	3.0726 ± 1.0440
KRR	0.6601 ± 0.2990	-0.5734 ±1.2856	1.2287 ± 0.8725	2.9507 ± 0.7882

3.2 Naive Model: Ordinary Least Squares (OLS)

Our initial baseline model was constructed from ordinary least squares (OLS) regression using all of the available 1,862 features. Because we only have 34 training points and we need to assign "weights" to 1862 features, our regression problem is underdetermined; i.e. there are infinitely many solutions to assigning weights to attain a zero-error training model. The perfect training metrics and poor validation metrics are indications of severe overfitting (Figure 3). In other words, the model can achieve a low error by fitting fluctuations in the training data that may not be representative of the true distribution, but this makes it a poor predictor of validation data.

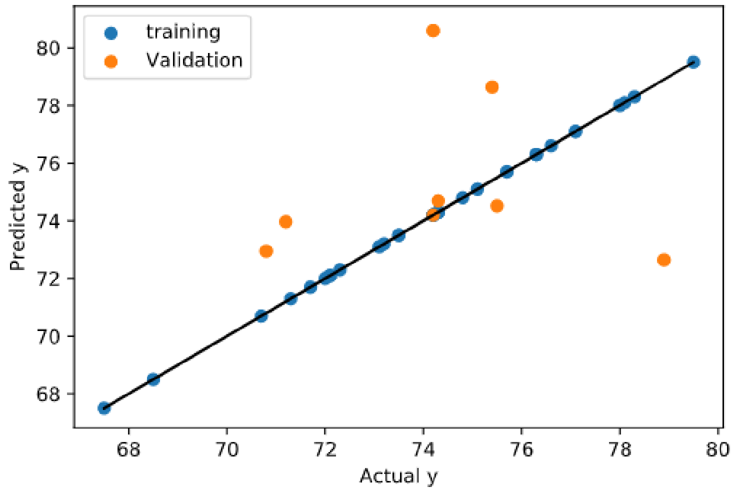


Figure 2: Parity plot for Ordinary Least Squares Regression model using `np.random.seed(40)`.

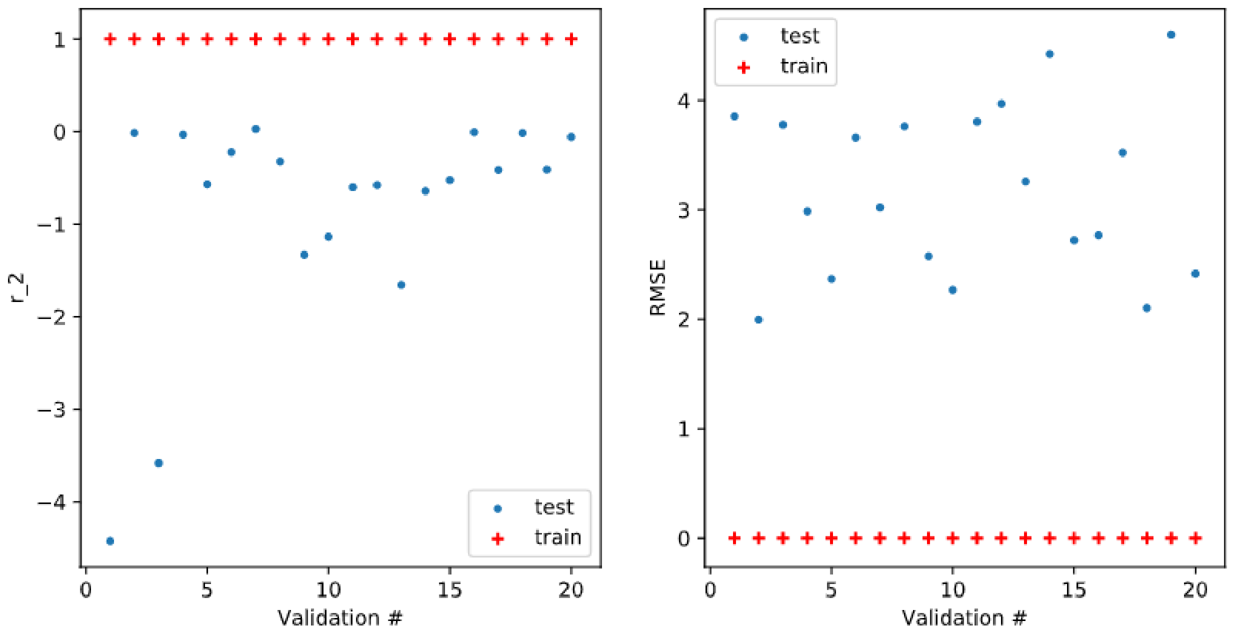


Figure 3: R^2 and RMSE results for the OLS model using 20 different training/validation hold-out splits.

3.3 Baseline Model: Principal Component Regression (PCR)

One strategy for combating this underdetermined system is to transform the feature space into a set of principal components which can "explain" the variance in feature values. By using a small number of principal components ($n_{comp} < 34$), we can come up with a unique solution to our original linear regression problem. We used `GridSearchCV()` with leave-one-out cross validation to determine the optimal number of principal components for regression. For `np.random.seed(40)`, we found that applying a linear regression model with 10 principal components yields the lowest RMSE values (Figure 5). However, depending on the hold-out split, the number of components that provides the lowest RMSE may change because our dataset is so small.

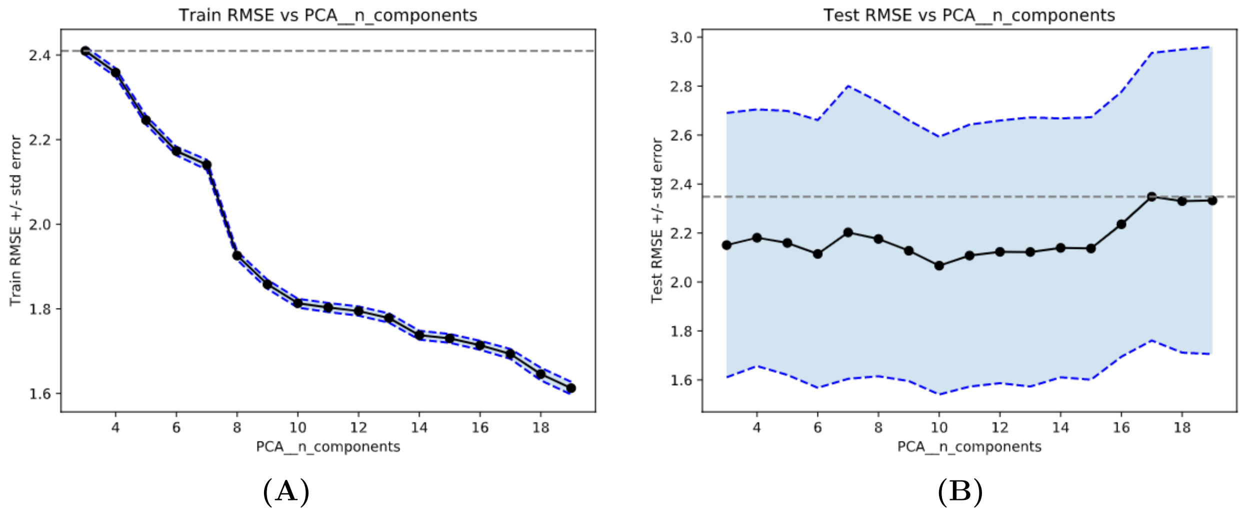


Figure 4: (A) Training RMSE plotted as a function of the number of principal components for linear PCA. (B) Testing RMSE plotted as a function of the number of principal components for linear PCA using `np.random.seed(40)`.

As expected, the RMSE in the training data monotonically decreases with the number of principal components (Figure 4). This trend can be explained by considering that the principal components capture variance explained in the training dataset, so including more components better explains the training data. However, PCA only considers the feature space in determining variance and ignores covariance between features and outputs. On the contrary, the testing RMSE appears to be relatively constant even when increasing the number of components. This RMSE invariance could be because the validation dataset's variance is not well-explained by the principal components of the training dataset.

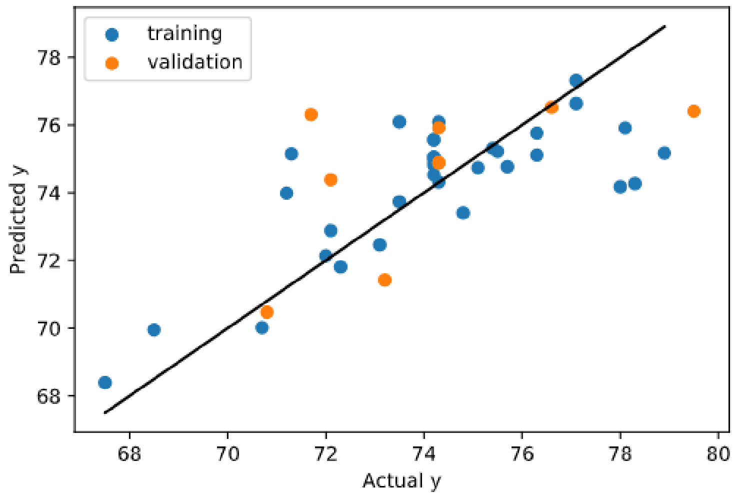


Figure 5: Parity plot for Principal Component Regression model using `np.random.seed(40)`.

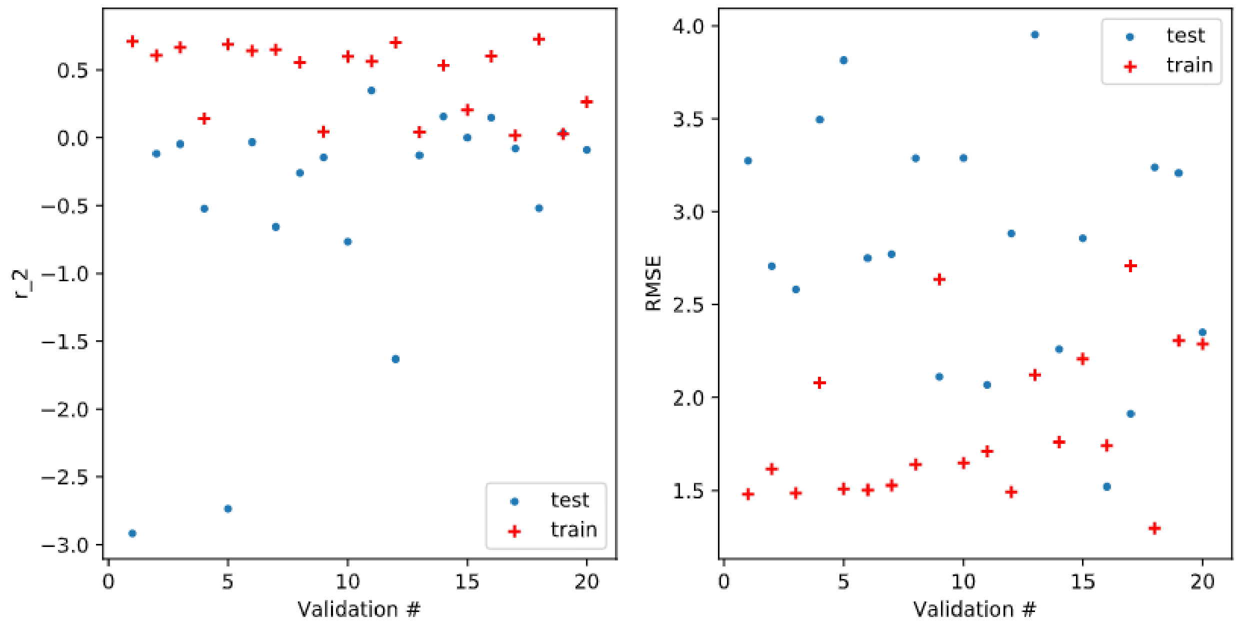


Figure 6: R^2 and RMSE results for the PCR model using 20 different training/validation hold-out splits.

When we cycle through different validation seeds (Figure 6), we notice that the overall performance of PCR on the training dataset is quite poor because the covariance between outputs and features are not considered.

3.4 Improved Model 1: Partial Least Squares Regression (PLSR)

Our first improved model applies partial least squares regression (PLSR). A PLSR model may be more effective than a PCR model because the latent variables (number of components) are calculated by maximizing the covariance in both $\mathbf{X}$ and $\mathbf{y}$. Additionally, it could be an appropriate choice when the dimensionality of the predictor space is larger than the number of observations, as is the case here. The results of the PLSR model are shown in Figures 9. Similar to the baseline PCR model, the RMSE decreases as the number of components increases for the PLSR model. A key difference is that only 4 or 5 components were found to give the optimal results with PLSR, compared with 10-17 components for the baseline PCR model.

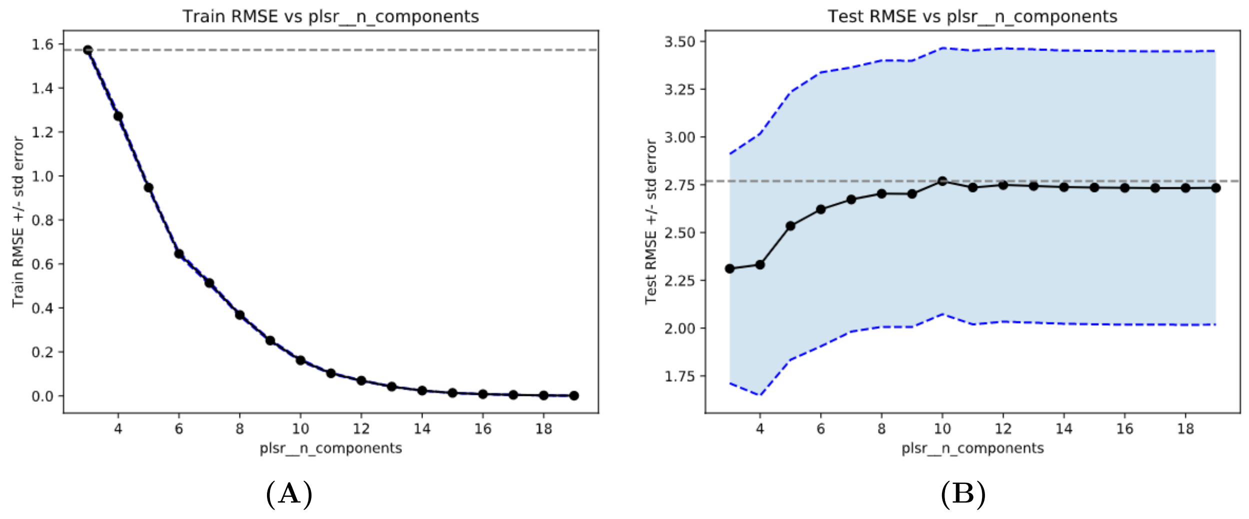


Figure 7: (A) Training RMSE plotted as a function of the number of components (latent variables) for PLSR. (B) Testing RMSE plotted as a function of the number of components (latent variables) for PLSR using `np.random.seed(40)`.

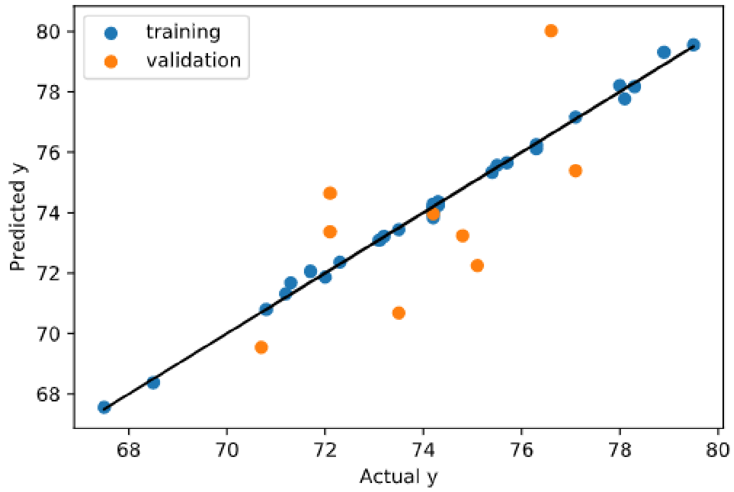


Figure 8: Parity plot for PLS regression model using `np.random.seed(40)`.

Initially, we hypothesized that PLSR would outperform PCR because PLS weights high variance directions in $\mathbf{X}$ that correlate with $\mathbf{y}$. This would remove cases where we have high variance $\mathbf{X}$ components with little $\mathbf{y}$ variance. However, we see that the empirical performance of PLSR and PCR are similarly poor, displaying low R^2 values (validation R^2 averages: -0.4981, PCR; -0.5364, PLSR). This poor performance could be potentially attributed to the deficient size of the dataset.

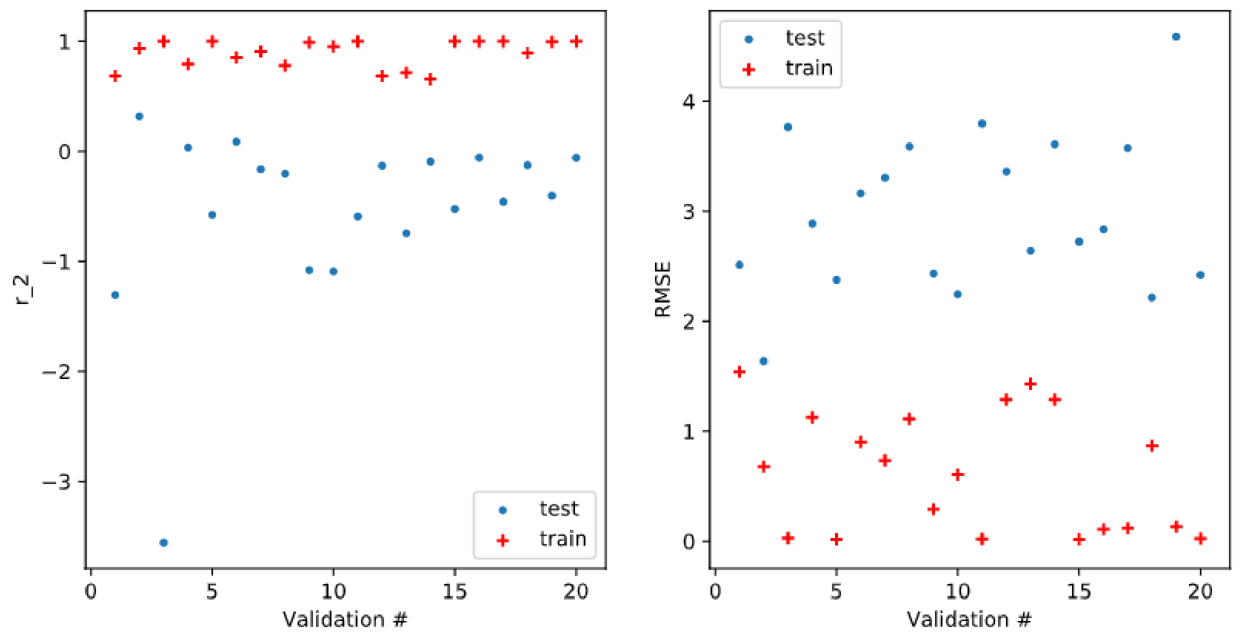
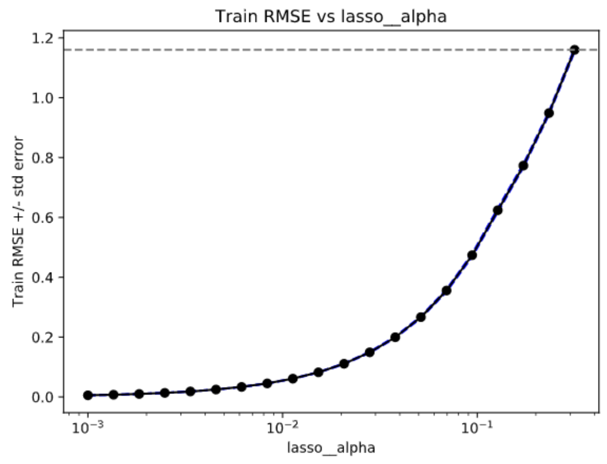


Figure 9: R^2 and RMSE results for PLSR model using 20 different training/validation hold-out splits.

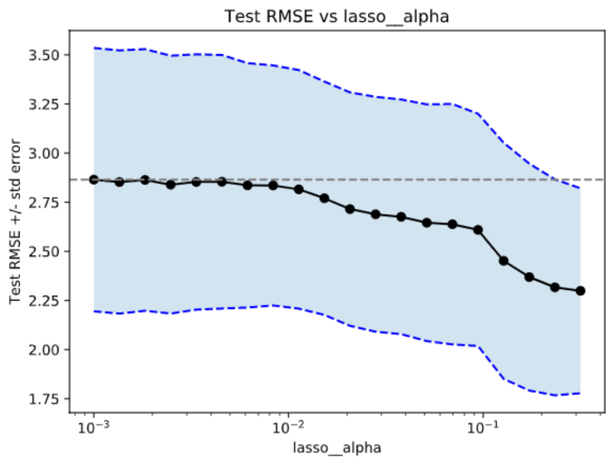
3.5 Improved Model 2: LASSO

Our second and third improved models use a different strategy than dimensionality reduction. Instead of defining components which contribute the most variance, we aim to constrain the coefficients that define the feature weights to reduce the risk of overfitting (regularization).

In this improved model iteration, we employ L_1 regularization (LASSO), where the regularization hyperparameter α was optimized using `GridSearchCV()` in search of minimized RMSE. The α values were chosen across a range of several orders of magnitude ($\alpha \in [10^{-3}, 10^{-0.5}]$). In 20 different iterations of `train_test_split()`, the most frequent best parameter is $\alpha = 0.001$, because the scoring metrics is minimized RMSE. At really small regularization strengths ($\alpha \leq 10^{-2}$), the RMSE for the training data is close to 0, which yields results similar to the baseline linear regression model without any regularization. The upper bound for α during the hyperparameter tuning was also chosen to avoid reducing all of the model parameters to 0 and therefore producing a trivial model. Increasing regularization generally increases the maximum error attained by the model. This trend makes intuitive sense because high α values increase the feature sparsity, meaning that fewer features are used for prediction.



(A)



(B)

Figure 10: (A) Training RMSE plotted as a function of hyperparameter α for LASSO. (B) Testing RMSE plotted as a function of hyperparameter α for LASSO using `np.random.seed(40)`.

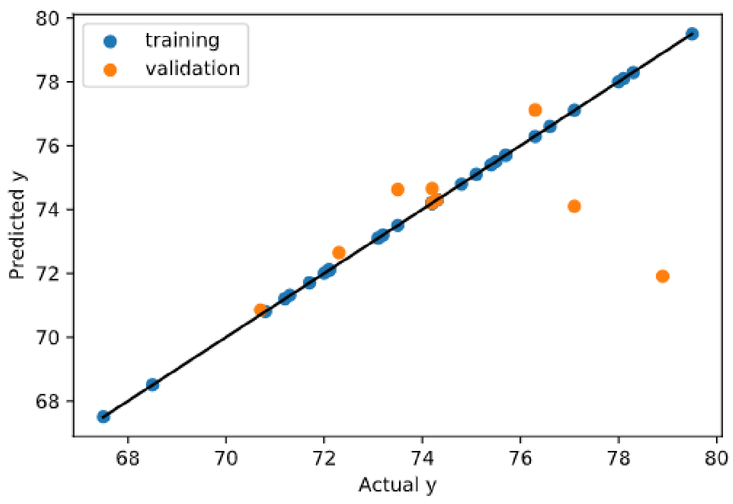


Figure 11: Parity plot for LASSO regression model using `np.random.seed(40)`.

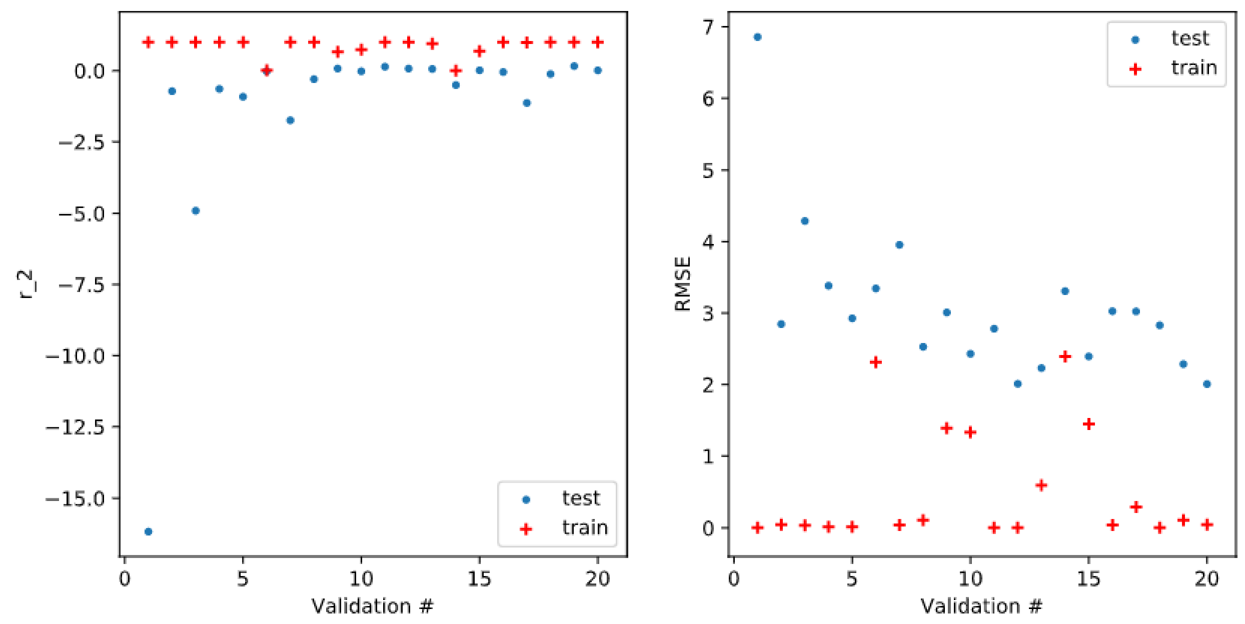


Figure 12: R^2 and RMSE results for LASSO model using 20 different training/validation hold-out splits.

In a broader range of α values, training RMSE values further increase with increased α, due to lost information at stronger regularization. However, RMSE test values (during cross validation) would further decrease as α increases (Figure 13), indicating an overfitting. The observed plateau after $\alpha = 10^1$ is a result of trivial solution where all features are wiped out by regularization. The LASSO model converges under maximum 10,000 iterations and a tolerance of 0.0001.

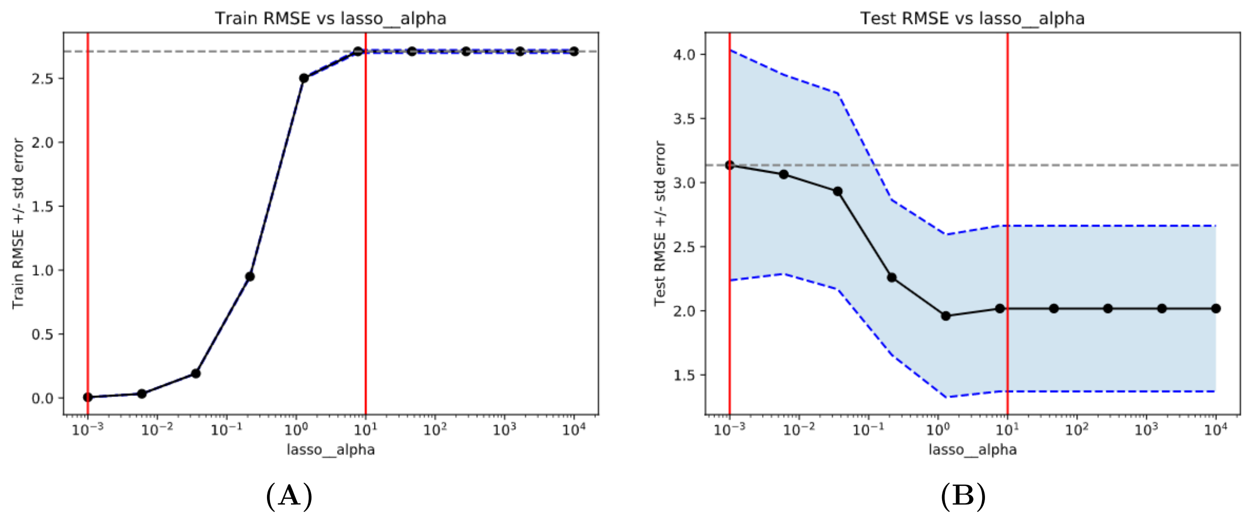


Figure 13: Convergence test for LASSO model on a broader range of hyperparameter α. Region in between the red lines are the region to avoid trivial solution. (A) Training RMSE increases as α increases the regularization strength. (B) Test RMSE decreases as α avoids some overfitting.

Compared to the OLS baseline model, the LASSO regression model showed slightly better validation

RMSE values, but the standard deviations are quite high when considering the 20 different training/validation sets (Table 1). The LASSO regression model could potentially be more advantageous as it does not overfit the training data and can possibly show better performance with different random seed values. Additionally, since LASSO regularization can set some of the model coefficients to 0, the model can provide information about which molecular descriptors of the additives are more impactful for ultimately affecting the fiber tenacity.

3.6 Kernel Ridge Regression

Finally, we implemented an L_2 regularization model using Kernel Ridge Regression (KRR). In our dataset, the number of features is much larger than the observations n. Subsequently, L_1 regularization can at most select n nonzero features, due to the nature of the convex optimization problem. To overcome this drawback, we hypothesize that the use of L_2 regularization may be beneficial in retaining more information present in the features while still applying a penalty to the regression weights.

Using our pipeline, `GridSearchCV()` was able to find the optimal hyperparameters α and γ for the KRR model for 20 different training/validation hold-out splits. In order to visualize the effect of the different hyperparameters on the RMSE values, we used the average value of α or γ while changing the other hyperparameter (Figure 14 and 16). The value of α controls the strength of L_2 regularization, where a higher value forces the model coefficients to be smaller and enhances the smoothness of the model. As $\alpha \to 0$, the model will begin to resemble OLS and likely lead to over-fitting. Both extremes are undesirable and lead to higher test RMSE values (Figure 14). As seen in Figure 14, the lowest testing RMSE is observed for $\alpha \approx 10^{-1}$, but there is a wide range of α values where the test RMSE is low, suggesting that there may be several possible values for the α that could provide suitable L_2 regularization.

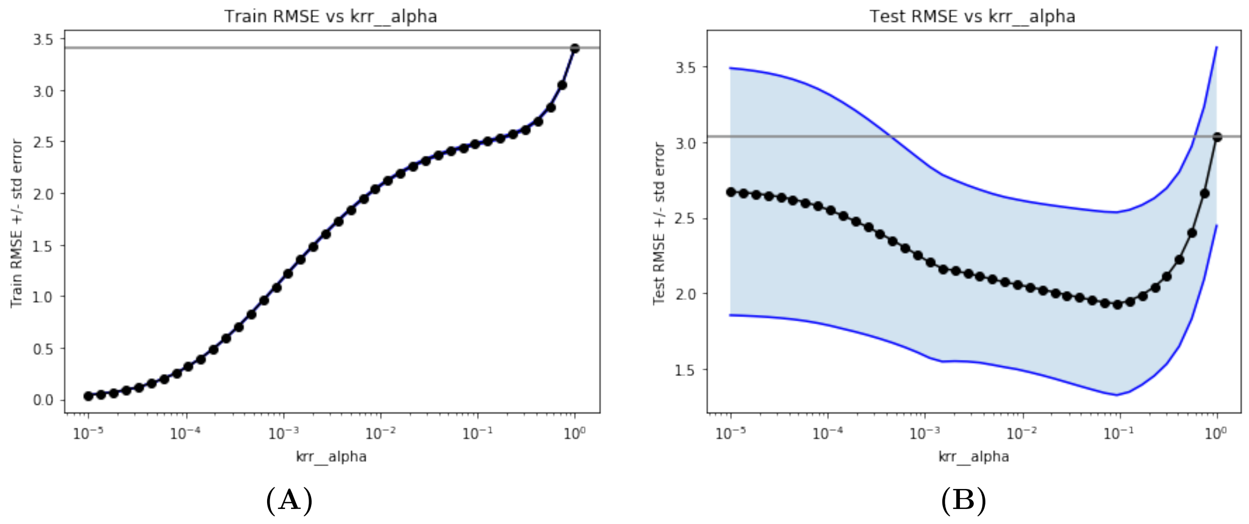


Figure 14: (A) Training RMSE plotted as a function of hyperparameter α for KRR (fixed $\gamma = 1.34 \times 10^{-6}$, average hyperparameter value from 20 different training/validation splits). (B) Testing RMSE plotted as a function of hyperparameter α for KRR using `np.random.seed(40)`.

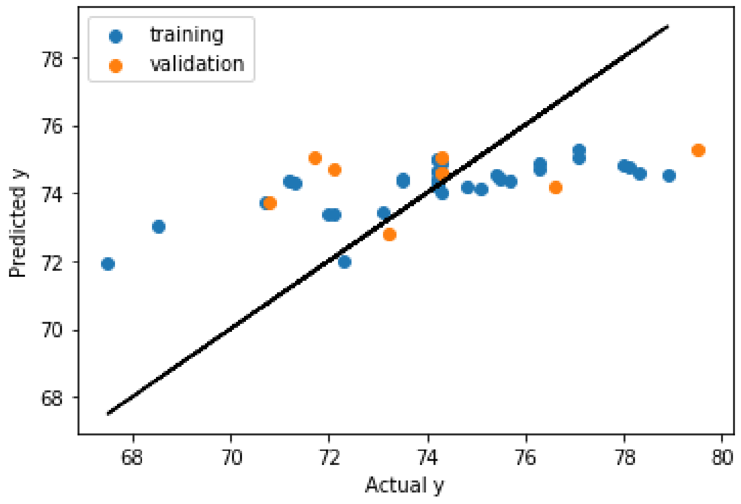
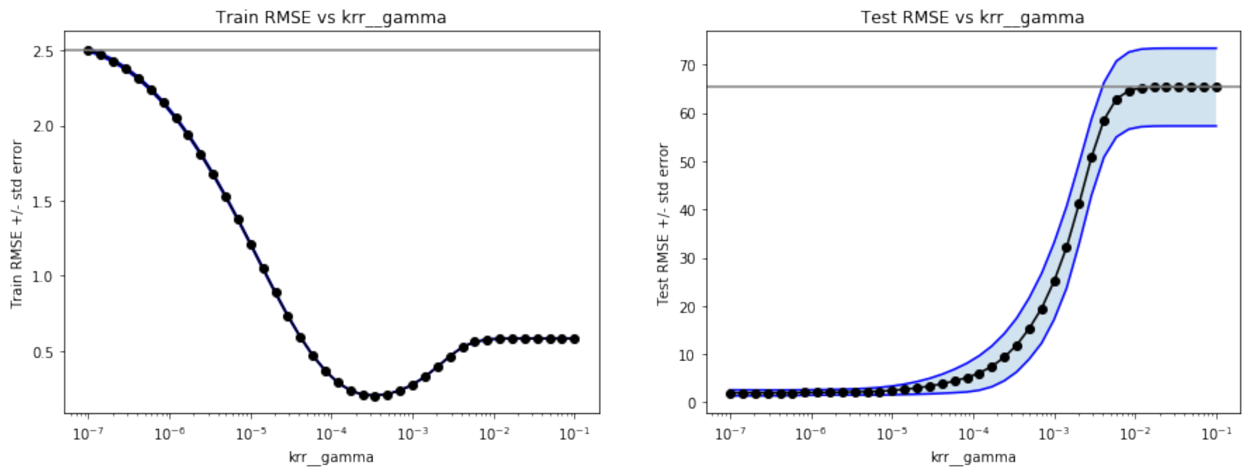


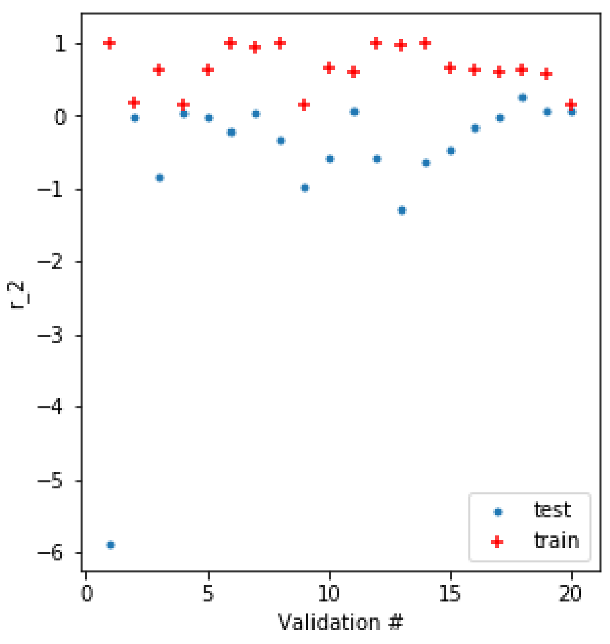
Figure 15: Parity plot for KRR model using `np.random.seed(40)` and average hyperparameter values $\alpha = 8.25 \text{ x } 10^{-3}$ and $\gamma = 1.34 \text{ x } 10^{-6}$ found from 20 different training/validation hold-out splits.

The γ hyperparameter controls the width of each Gaussian function in the `rbf` kernel, where large γ values correspond to more narrow curves and vice versa. As shown in Figure 16, smaller values of γ lead to lower test RMSE values. This observation makes sense for our sparse dataset, since lower γ values have much wider Gaussian curves, and as a result, each training point has more influence on the model. While there is noticeable minimum in the training RMSE for γ between 10^{-4} and 10^{-3}, the average optimal γ value for the 20 different training/validation splits was on the order of 10^{-6} (Figure 15).

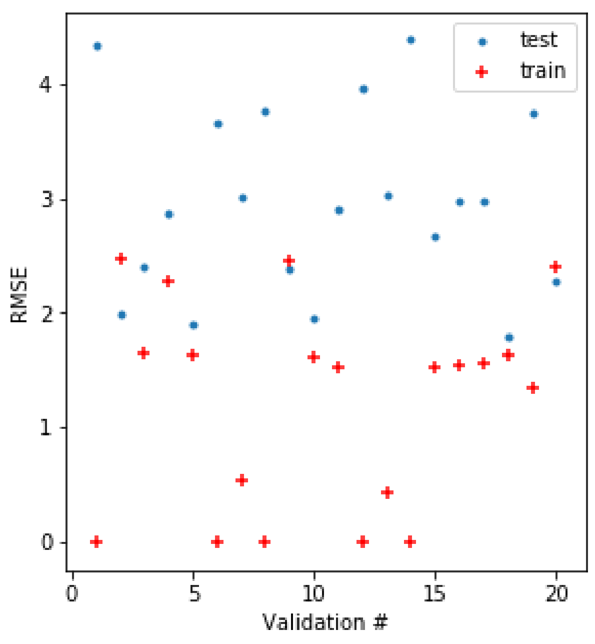


(A) **(B)**

Figure 16: (A) Training RMSE plotted as a function of hyperparameter γ for KRR (fixed $\alpha = 8.25 \text{ x } 10^{-3}$, average hyperparameter value from 20 different training/validation splits). (B) Testing RMSE plotted as a function of hyperparameter γ for KRR using `np.random.seed(40)`.



(A)



(B)

Figure 17: R^2 and RMSE results for KRR model using 20 different training/validation hold-out splits.

4 Conclusions

In this report, we have applied several linear models to draw relationships between various additives' chemical structures and the resulting thermal stability of the composite PET fibers. Our initial baseline OLS model showed overfitting and poor validation performance. The PCR model reduced the dimensionality of the dataset while improving the validation performance. Our initial linear improved models in LASSO and PLS did not show significant improvement when considering 20 different training/validation sets. For the improved model using KRR, the hyperparameters were also tuned for 20 different training/validation sets, but the resulting average α and γ values did not lead to a significantly improved model. In general, it was observed that linear regression works best when the underlying model exhibits linear correlations. The results presented here for the fully-linear models suggest that there may be non-linear correlations inherent in the dataset.

Overall, the small size of the dataset proved to be a key bottleneck in the performances of our models. Many models were shown to overfit, thus giving poor prediction on validation data despite a decent scoring metric during training. A non-linear models such as ANN can be a powerful universal approximator, but our implementation was held back by the small number of data points (Appendix B). The small number of data points also prohibited us from adopting feature selection techniques such as the Bayesian Information Criterion (BIC) for LASSO.

We propose three different ideas for analyzing this dataset in the future:

- **Include more data.** We can gather more data via data-mining, provided there are other patents or literature studies that were performed under similar conditions (though this is likely a time-consuming exercise). Alternatively, we can supplement the existing dataset with synthetic data using a generative model. Generative classification models such as Gaussian Mixture Model (GMM) can also be adopted to generate synthetic data to help aid training.

- **Re-formulate the problem.** Predicting good performance with regard to tenacity persistence may be better suited for comparing a select number of additives that belong to a set of clusters (with each cluster representing similar mechanical behavior). It may be easier to develop a classification model where we aim to answer the question "Is this additive 'sufficiently' beneficial?" for screening purposes. Hypothetical molecules could be transformed onto the descriptor space to determine whether or not they fall into the "good performing" classification regions prior to testing them in the laboratory.

- **Alternative feature representations for molecular candidates** While the molecular descriptors used in this dataset were not calculated as part of this project, they represent just one example of how molecules can be featurized. While there is a significant number of descriptors included, only domain knowledge could help determine whether or not the descriptors are actually chemically representative of the tenacity persistence of the molecules. Other representation methods can also be considered, such as MACCS fingerprinting (which uses 166 different chemical keys to answer questions about the molecule that are more interpretable than the descriptors in this work), or other descriptor-based approaches could potentially be better suited for representing the data here. These alternative methods could potentially be worth trying for the sake of building a regression model or classifier.

5 Individual Contributions

5.1 Summary

Our group met in-person/virtually 1-2 times per week to work on the project and delegate coding and report duties for each assigned milestone. All of our group submissions were initially based on Google Colaboratory, and transferred to Jupyter notebooks for submission. For code development for the final model, we initially used Brian's code as a basis for its streamlined Pipeline framework. The linear models provided in the final report are equal contributions from all group members. Nicole developed the artificial neural network model (Appendix) and scripts for running multiple validation splits, as well as formatting the final report in LaTeX. Max also developed a KRR model for the final report.

5.2 Itemized List of Contributions

Name Keys: [MB] - Max Bukhovko; [NH] - Nicole Hu; [BK] - Brian Khau; [AL] - Aaron Liu; [RV] - Rahul Venkatesh; [E] - Everyone

1. Project Selection
   - Dataset selection; [AL]
   - Document preparation; [E]
2. Baseline Model
   - Approach brainstorming/Workflow development [E]
   - Coding:
     - Dataframe import, pre-processing [AL]
     - Visualization helper functions [AL, RV]
     - PCA helper functions [NH, BK, RV]
     - Parity plots [MB, BK]
   - Report:
     - Context [BK, AL]
     - Objectives and dataset details, approach [BK, AL]
     - Summary [NH, AL]
3. Improved Model
   - Pipeline Implementation [BK]
   - Princpal Component Regression [AL, RV, MB]
   - Partial Least Squares [BK]
   - LASSO [AL, RV, MB, NH]
   - KRR [MB]
   - Artificial Neural Network Implementation [NH]
   - Code Integration and Seeding Routines [NH, MB]

4. Report Draft and Final Report
    - Introduction [AL]
    - Methodology [RV, AL]
    - Block flow diagram [BK]
    - Results [NH, BK, MB]
    - Discussion [E]
    - Conclusion [MB, AL, BK]
    - Tables and figures [RV, NH]
    - Appendix [RV, NH]

6 Appendix A: Random seed results

Model	Hyperparameter	Training R^2	Validation R^2	$\text{RMSE}_{\text{train}}$	$\text{RMSE}_{\text{valid}}$
OLS	N/A	1.0000	-0.2535	0.0000	2.2582
PCR	17 components	0.5602	0.2525	1.7230	2.1755
LASSO	$\alpha = 0.001$	1.0000	-0.2096	0.0056	2.5858
PLSR	10 components	0.9960	-0.1622	0.1704	2.1745
KRR	10 components	0.9960	-0.1622	0.1704	2.1745

Table 2: Calculated metrics when the hold-out split is determined by `np.random.seed(40)`.

7 Appendix B: Artificial Neural Network (ANN)

- `Pipeline = StandardScaler(), PCA(), Sequential()`

In ANN, two hyperparameters, learning rate and number of epochs for training are identified and optimized. In Figure 19, loss did not further increase with number of epochs increased, suggesting an unconverged model (ANN, single layer of 5 nodes). Further analysis also suggests that ANN model did not converge for this particular data set (Figure 20 and 21). Different activation functions and NN architectures were also tested. Tested combinations (ReLU + 2 layers + 5 - 20 nodes per layer, tanh + 1 layer + 5 - 20 nodes) seemed to be limited by the small number of sample points and did not result in a high performance regression on validation.

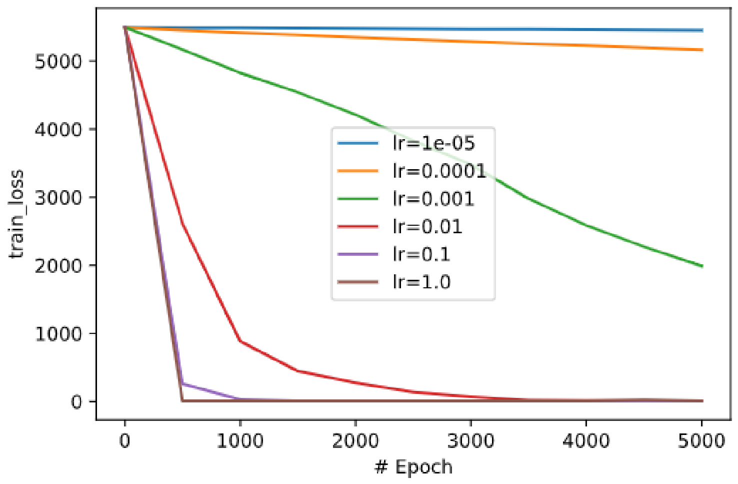


Figure 18: Learning rate optimization.

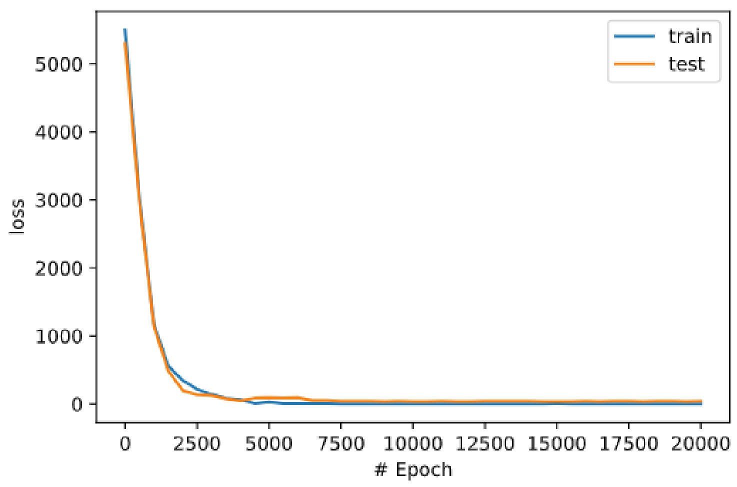


Figure 19: Number of epochs optimization to avoid overfitting.

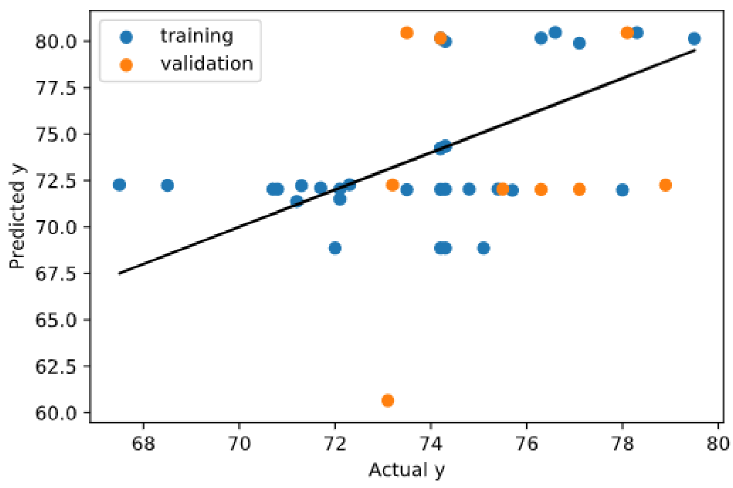
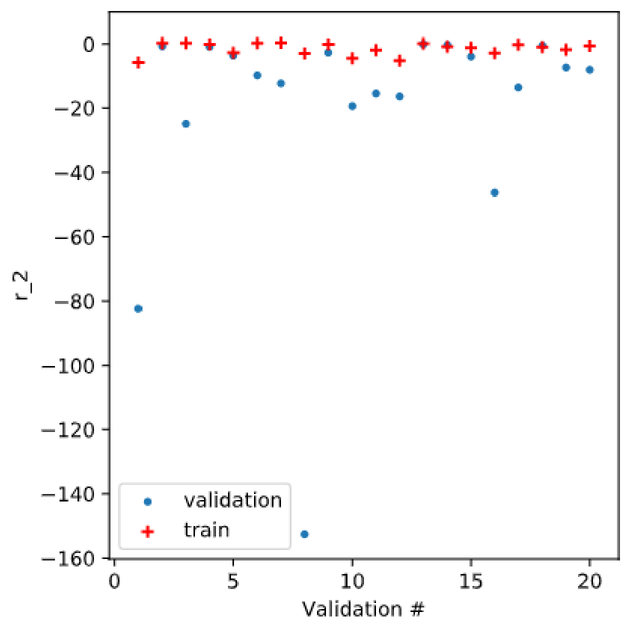
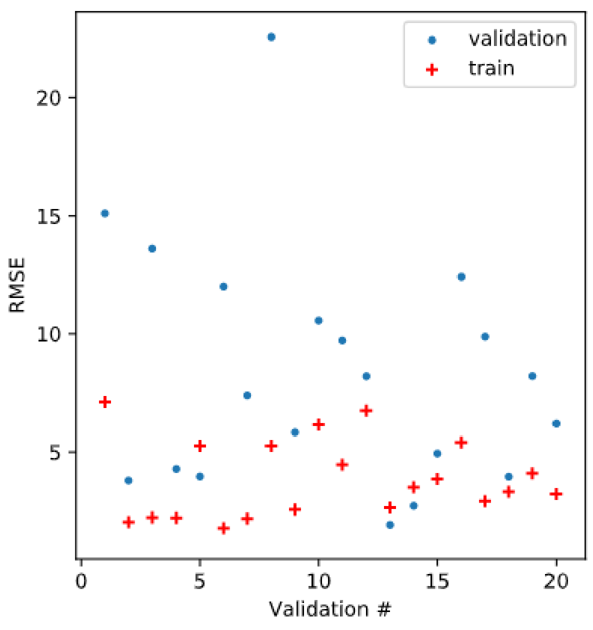


Figure 20: Parity plot for ANN regression model using `np.random.seed(40)`



(A)



(B)

Figure 21: R^2 and RMSE results for ANN model using 20 different training/validation hold-out splits.

8 Appendix C: Corresponding Scripts

Scripts used to produce the results and graphs in this report is zipped in the same folder as this report.
To break the contents down:

1. OLS, PCR, LASSO, PLSR (`Group_8_final_models.ipynb`)
2. KRR (`Group_8_KRR model.ipynb`)
3. Data visualization (`Group_8_final_visualization.ipynb`): correlation heat map, feature distribution