Data-Driven Design of a Supercompressible Meta-Material

Brown University
ENGN2350 Data-Driven Design and Analysis of Structures and Materials

Course Instructor: Miguel A. Bessa
Submitted by: Chenjie Gan, Justin Miller
Submitted on: December 7, 2023

Overview

For the final project of the Data-Driven Design and Analysis of Structures and Materials (3DASM) course, the group will work with an existing dataset to model and predict the behavior of a variably coilable structure and try to find optical design with critical buckling stress and energy absorbed. This involves analyzing and drawing conclusions in a similar manner to the original "supercompressible" paper from Bessa et. al[1]. This type of work is extremely relevant in the current technological atmosphere as machine learning and artificial intelligence is becoming more and more intertwined with traditional engineering problems.

This particular project has multiple goals. The primary objective is to develop both classification and regression models for two different datasets, comparing different types of algorithms on different types of tasks to identify where (and why) some perform better than others. The group achieved **97% accuracy** score for classification and **0.999 R-squared** score for regression. Secondary objectives are to use hyperparameter optimization to create the best possible model and to use optimization within the feature space to find the best possible input features to result in strong and springy structures that maintain coilability. The group achieved a design with critical bulk stress of **118 KPa**, and a design with absorbed energy of **48 KJ per cubic meter**.

All of these components involve understanding both the data involved in the problem and the physical basis that the problem is trying to uncover. In general, a theme of this project (and course) is to combine the theoretical and practical aspects of different machine learning models to compare their applicability to problems in solid mechanics. In the pages to come, the project group aims to do this by discussing in depth examples of how both Bayesian and Deterministic models apply to mechanics problems, and how they can be used appropriately to generate answers coinciding with engineering truths.

Content

1. Data characterization

a. Data and Bounds

To contextualize the problem, it is important to first summarize the data availability and bounds. The 3 dimensional data set has features defining the ratios of the structure's diameter (ratio_d), pitch (ratio_pitch), and top diameter. (ratio_top_diameter). Bounds and data count of each 3-d input variable is shown in Table 1.

Table 1. 3 Dimensional Input Metadata

Input Feature	Min Bound	Max Bound	Available Data	Missing Values
ratio_d	0.0040	0.0729	1000	0
ratio_pitch	0.2500	1.4988	1000	0
ratio_top_diameter	0.0000	0.7992	1000	0

The same information for the 7-d dataset is shown below in Table 2.

Table 2. 7 Dimensional Input Metadata

Input Feature	Min Bound	Max Bound	Available Data	Missing Values
ratio_area	1.170e-5	0.0041	50000	0
ratio_Ixx	1.128e-11	1.400e-6	50000	0
ratio_ Iyy	1.128e-11	1.400e-6	50000	0
ratio_J	1.353e-11	7.770e-6	50000	0
ratio_pitch	0.2500	1.5000	50000	0
ratio_top_diameter	0.0000	0.8000	50000	0
ratio_shear_modulus	0.0350	0.4500	50000	0

Both the 3-d and 7-d dataset have three output features: structural coilability (coilable), critical stress (sigma_ctir) and energy (energy). The data count of each output variable for each dataset is shown in Table 3.

Table 3. Overall Output Data Counts

Output Feature	3D: Available	3D Missing (NaN)	7D Available	7D Missing (NaN)
coilable	1000	0	50000	0

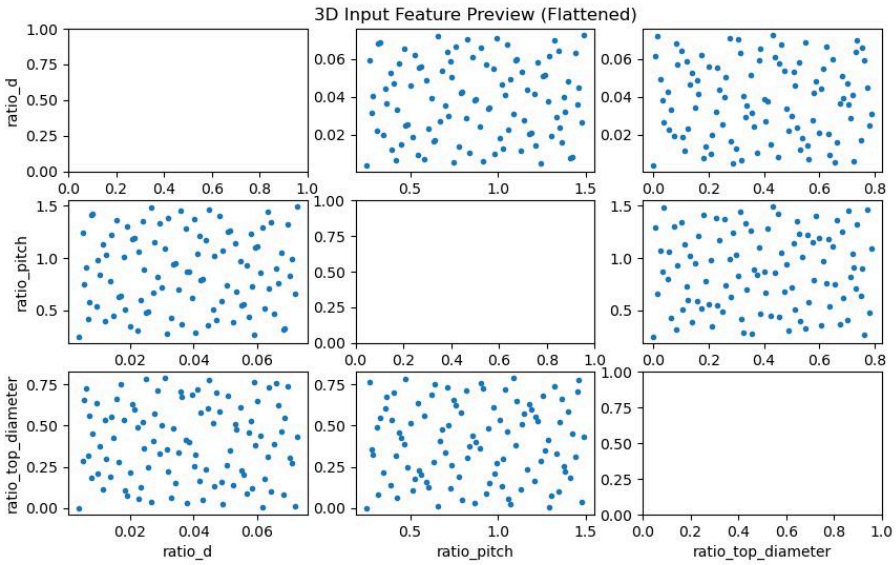
sigma_crit	949	51	49167	833
energy	524	476	22564	27436

Table 3 reveals a significant chunk of missing data. This fact will be important later in model analysis, particularly when faced with the optimization task in Section 4.

b. Input Features

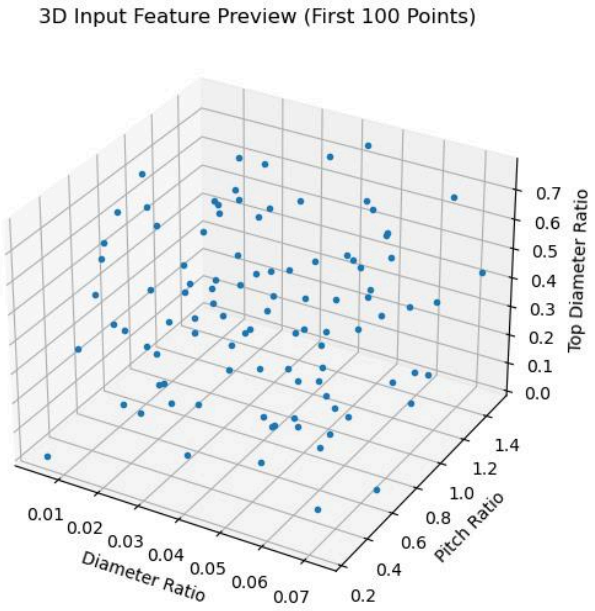
Figure 1 shows the distribution of experimental data to give some context on the approach. As you can see, there is a pattern to the 2d input data among all 6 combinations of features. This experiment was designed to span all areas of the bounded input space, making sure that all possible combinations of these parameters were represented for the machine learning model to learn from.

Figure 1. 3d Input Feature Preview (Flattened)



Visualizing this in 3D, there exists the patterned characteristic of the sampling method persists between dimensions as well. This Sobol sequence is not completely random, as it wants to capture all areas of the feature space. This results in a minor zigzagging pattern across the data so it captures sufficient variation. A reduced 3-d plot shows the first 100 points in this pattern in Figure 2.

Figure 2. 3D Input Feature Preview



c. Output Features

To examine the bounds and patterns of the output features, histograms were created for both the 3 Dimensional and 7 Dimensional data sets in Figures 3 and 4, respectively.

Figure 3: 3D Dataset Output Histograms

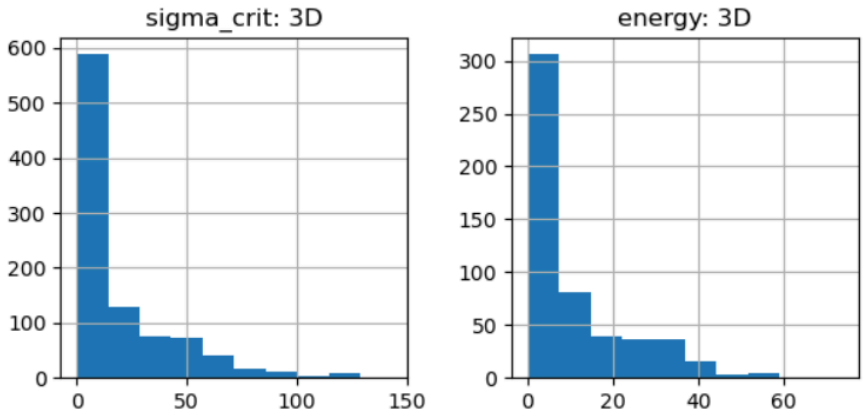
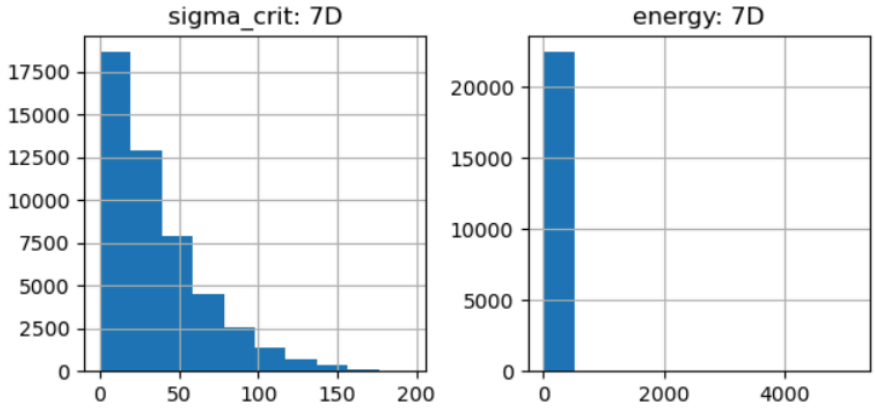


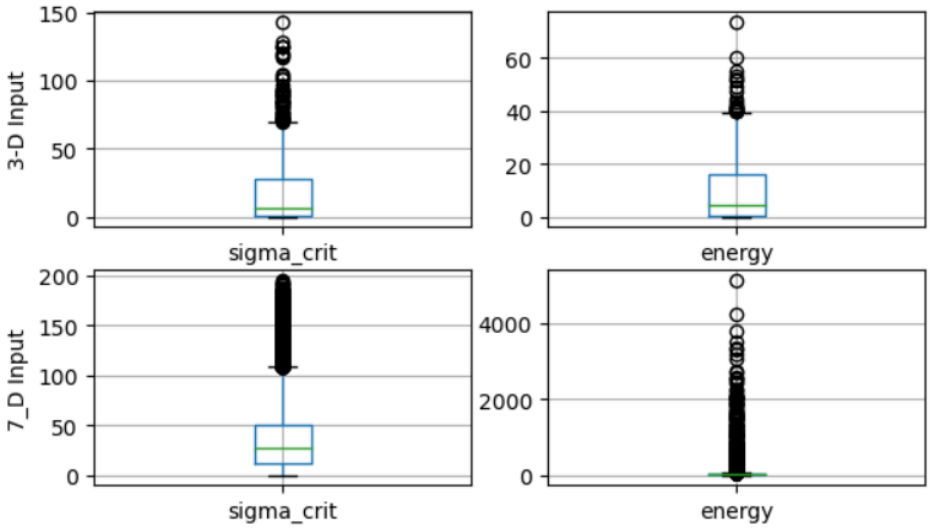
Figure 4. 7D Dataset Output Histograms



From the above figures, one sees all output data exhibits a right skew, with the skew being severe for energy on the 7 Dimensional data sets. To compensate for the weaker skews, the group will use a Standard Scalar to ensure effective training of the machine learning algorithms. While the outliers of 7D energy are more significant, it is too rash to remove those straight away. They are not necessarily erroneous values, it is possible that the finite element analysis (FEA) simulation used to generate the values uncovered a true phenomena. However, it may be useful later in the process to look further into these values, potentially applying some sort of transformation (e.g. logarithmic) in addition to the standard scalar to help the model train.

To give a better idea of outliers and how those might affect any scaling, Figure 5 shows box and whisker plots for the histograms above.While this representation is not specifically asked for in the project write up, it served as useful to the group since it provides insight into differences in model results for each dataset, particularly on the energy variable.

Figure 5. Output Boxplots



A similar analysis listing the count of the categorical variable is shown in Table 4 to contextualize the output space of structural coilability.

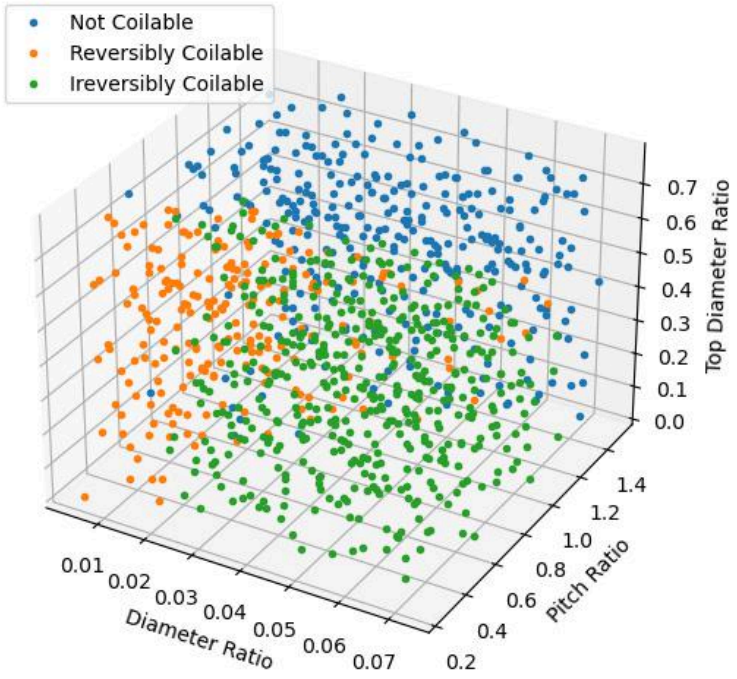
Table 4. Output Category Distribution

	Not Coilable	Coilable	Coilable but Yields
3 Dimensional Dataset	318	214	468
7 Dimensional Dataset	17669	32331	N/A

Since it is only a few categories there aren't any true "outliers", but the existence of a third class (coilable but yields) for the 3D dataset could make it difficult to compare results between datasets. As such, it will be beneficial to experiment with dropping the class entirely or combining it with pure "coilable" later in the project.

To get an overall sense of the output space, the group also generated a 3d view of the problem space colored by output class. This visualization (Figure 6), while not explicitly asked for in the write up, was very helpful in deciding which models to use and what results to expect. The visualization shows the separations of each class and relatively small noise between classes.

Figure 6. 3D Class Visualization



d. Key Values

As hinted in the project overview, a main purpose of the investigation is to identify key values to optimize the performance of the structure. In this situation, the definition of success is reversible coilability. However, depending on the situation there could be a preference towards either maximum critical buckling stress or energy absorption capability. These two key values for the 3 Dimensional dataset are shown in Tables 5 and 6.

Table 5. 3D Data Maximum Buckling Stress

Input Features	Input Values	**Output Features**	Output Values
ratio_d	0.071989	**coilable**	Reversibly Coilable
ratio_pitch	0.976318	**sigma_crit**	84.408673
ratio_top_diameter	0.477344	**energy**	NaN

Table 6. 3D Data Maximum Energy Absorption

Input Features	Input Values	**Output Features**	Output Values
ratio_d	0.033783	**coilable**	Reversibly Coilable
ratio_pitch	0.320801	**sigma_crit**	6.735183
ratio_top_diameter	0.757812	**energy**	3.732142

The impact of these input points as the so-called “best designs” will be further explored in Section 4 (Optimization).

2. Finding good machine learning models

To further examine the use of machine learning in meta-material design, this phase of the project will use 3 different algorithms for the same classification and regression tests contrasting the performance of the models on both the smaller 3-d dataset and full 7-d dataset. Furthermore, the group will discuss the theoretical considerations of deciding between models and touch upon the scalability of these models for more complex simulations. It is important to note that for both datasets 75% of the data was allocated for training while 25% was saved for testing, and that all 3-d models were run on 1,000 data points and all 7-d models were run on 50,000 data points.

a. 3-d Dataset

i. Classification: Three Classes

For the classification task, the primary model given in the project instructions is a **C-Support Vector Classifier**. This algorithm, commonly used in classification tasks, is known for its relative simplicity and strong performance on separable classification problems. Two additional algorithms were selected to compare with the SVC model. The first is **random forest**, which was selected due to its explainability and strong performance on decision based processes; the group expects this decision making strength to work well in metamaterial design. Random forest, which was not covered in class, is a bagging method which constructs multiple decision trees from different subsets of the data and outputs the class that receives the most consensus. The second is an **artificial neural network**, which was selected for both its relevance in modern research and its direct contrast with the simpler, more Bayesian SVC model. It is highly nonlinear and generally needs large datasets to yield good results, but it may be able to reveal patterns in the data that wouldn't be found by the two simpler models.

These classification models will be judged on quantitative and qualitative metrics. Quantitative will explore not just the accuracy (percent correct), but also the precision which gives the proportion of correct positives to false positives, the recall which gives the proportion of positives which were correctly identified, and the F1 score which finds the harmonic mean of those values to give a more general, single metric. Note that the precision, recall, and F1 are reported in respective vectors of length 3 for the 3-class classification problems as those metrics are specific to each class for non-binary problems. The qualitative consideration will be the simplicity of the decision boundary. Given similar performance, a simpler model is better than a more complex one. Along with accuracy and related scores the group will be looking for smooth boundaries and will try to avoid any signs of potential overfitting.

1. C-Support Vector Classifier

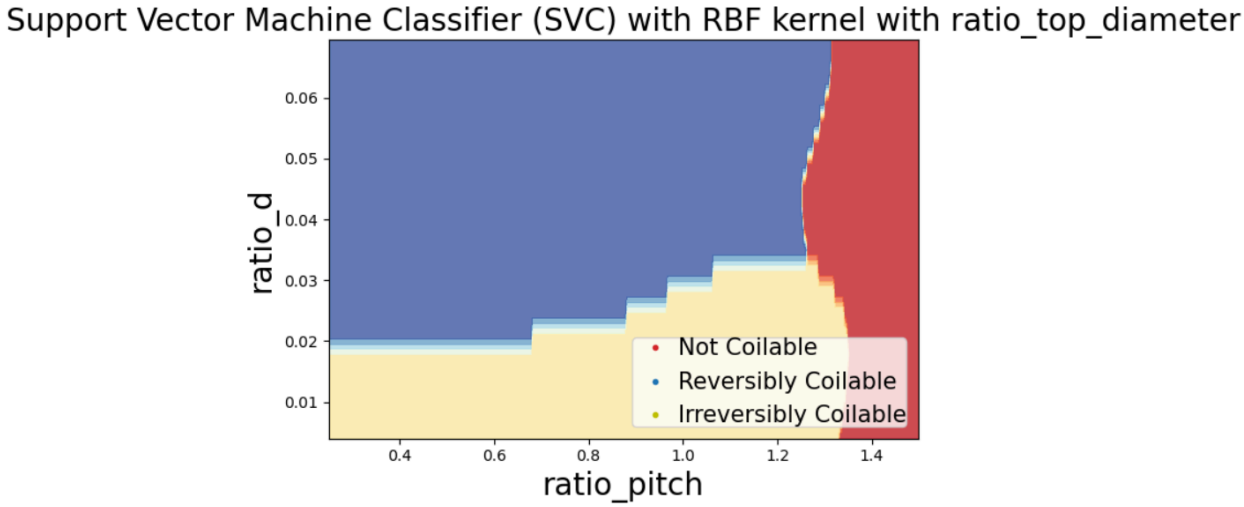
For this task, a C-Support Vector Classifier gave 85.2% accuracy. Further metrics for this classification success are shown along with the other models in Table 7. For now, the overall confusion matrix is shown below in Matrix 1.

Matrix 1. SVM Confusion Matrix

[72, 3, 3]
[2, 35, 16]
[4, 9, 106]

To gain an intuitive understanding of the resulting decision boundary of this method, a flattened plot is shown showing the model's prediction at an arbitrary ratio_top_diameter point (Figure 7). As you can see, the SVM results in a very smooth, simple decision boundary. The behavior of the model fits very well on this particular problem case since

Figure 7. Sample (Scaled) SVM Decision Boundary



To get this result, the SVM Model uses a Radial Basis Function kernel and the optimal hyperparameters were found using grid search. For the 3D model, these values were found to be C=1, gamma=0.9.

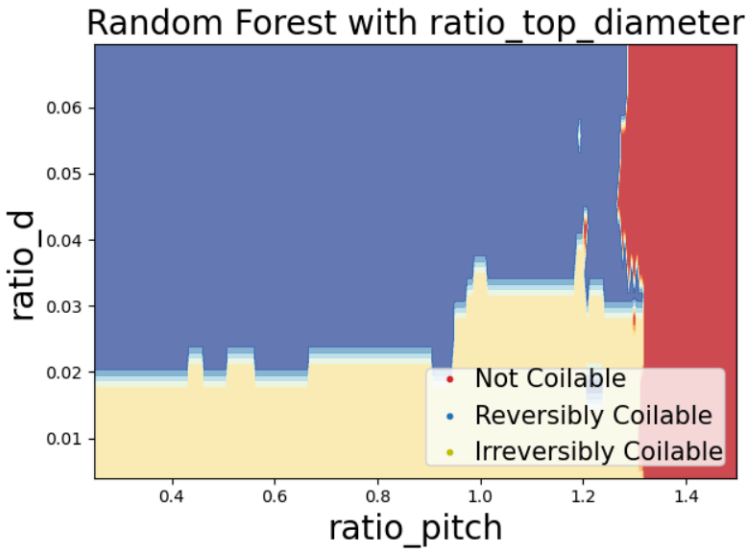
2. Random Forest Classifier

For this task, a Random Forest Classifier gave 85.2% accuracy. The confusion matrix and a 2d visualization of the decision boundary at an arbitrary third variable are shown below,

Matrix 2. Random Forest Confusion Matrix

[70, 5, 3]
[2, 34, 17]
[4, 6. 109]

Figure 8. Sample Random Forest Decision Boundary



As you can see, the random forest results in a much more complex decision space. While this model has a high accuracy, the choppiness of the feature space is not generally what is wanted for a problem with this level of simplicity. In the long run, using a model that looks like this will not be easily explainable and could be prone to overfitting.

Like the SVM, the Random Forest model uses optimal hyperparameters which were found using grid search. For the 3D model, these values were found to be n_estimators=200, max_depth=None, min_samples_split=2, min_samples_leaf=2.

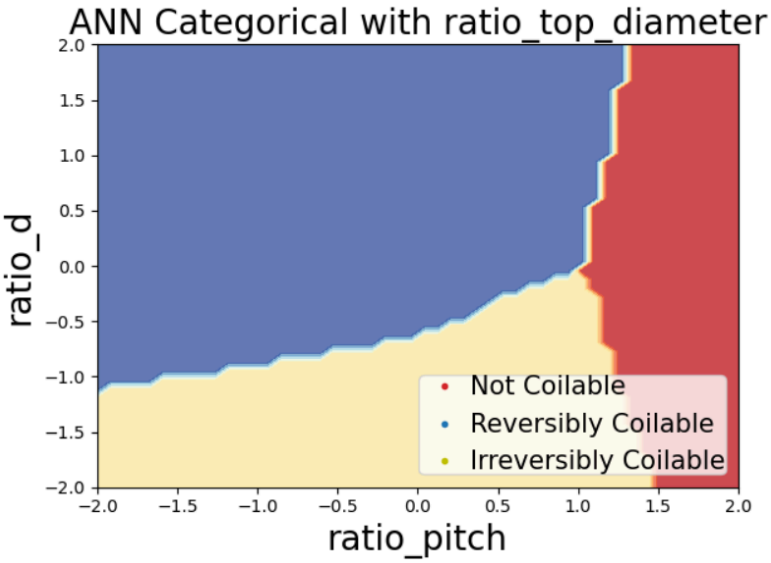
3. Neural Network Classifier

For this task, a Neural Network Classifier gave 82.0% accuracy. The confusion matrix and a 2d visualization of the decision boundary at an arbitrary third variable are shown below,

Matrix 3. Neural Network Confusion Matrix

[70, 6, 2]
[3, 32, 18]
[5, 11, 103]

Figure 9. Sample (Scaled) NN Decision Boundary



Unlike the prior classification models, the hyperparameters for the Neural Network were decided in an unscientific manner due to the complexity of the model. The final model was run with 64 neurons on the first hidden layer, 32 on the second, a ReLU activation function (with a sigmoid on the output layer), a batch size of 2500, 500 epochs, and an Adam optimizer with a learning rate of 0.001.

4. Overall Takeaways

In summary, the accuracy of these three methods for 3-class classification is shown in Table 7.

Table 7.3-d Classification Accuracy Metrics (3 Classes)

	Accuracy	Precision Vector	Recall Vector	F1 Vector
SVM (RBF Kernel)	0.852	[0.923, 0.745, 0.848]	[0.923, 0.660, 0.891]	[0.923, 0.700, 0.869]
Random Forest	0.852	[0.933, 0.733, 0.838]	[0.897, 0.623, 0.916]	[0.915, 0.673, 0.876]
ANN (two hidden layers)	0.820	[0.897, 0.653, 0.837]	[0.897, 0.604, 0.866]	[0.897, 0.627, 0.851]

Overall, these three models all have pretty average performances on the 3D classification with three classes. By performance, both the SVM and Random Forest have the highest accuracy.

However, the choppy visualization of the model's decision boundary indicates that it may be too complex for its own good, and it's important to recognize the value of SVM's much smoother decision boundary. In some situations, it makes sense to go with the more generalizable, explainable model even if the accuracy is a few points lower. Also with these results, the NN wouldn't even be in the conversation for the best model on this dataset as it is inherently more complex than both models yet still doesn't have the best accuracy

While there is room for disagreement, if the group were to choose 1 algorithm for 3 Class Classification on this dataset it would be the **Support Vector Machine** as it is simple and explainable while maintaining or surpassing the accuracy of more abstract models.

ii. Classification: Binary

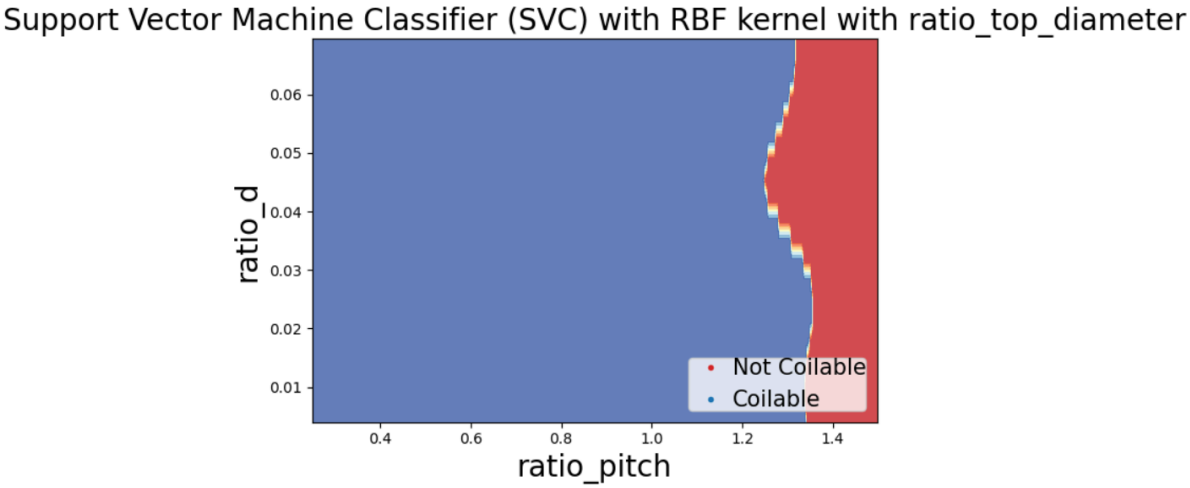
To better compare metrics with the 7-d dataset, this classification task was also repeated on a binary version of the coilable feature, where the "reversibly coilable" and "irreversibly coilable" classes are merged into a single class. This analysis also serves as a basis for considering the effects of output specifications on the performance of machine learning models on metamaterial classification.

1. C-Support Vector Classifier

For binary classification, the same C-Support Vector Classifier had an accuracy of 92.4%. The confusion matrix and a 2d visualization are shown below.

Matrix 4. SVM Confusion Matrix (2 Classes)
[69, 9]
[12, 160]

Figure 10. Sample SVC Decision Boundary Binary Classification (2 Classes)



It is obvious that the plot looks very similar to the 3 class SVC, just with the coilable classes merged. It is also clear that the decision matrix is the same (or very close) as it would be if one added the merged terms together as correct predictions. This result is expected as there were no major changes between the two models, just a merging of the classes, and these trends should continue for all models.

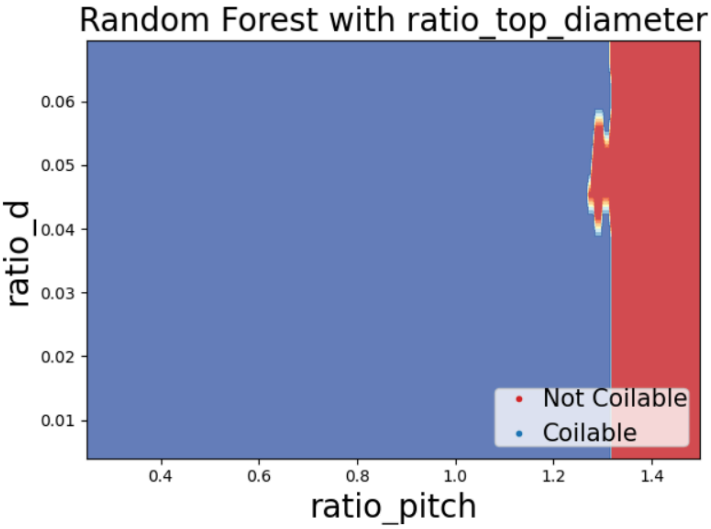
2. Random Forest Classifier

For binary classification, the same Random Forest Classifier had an accuracy of 96.0%. The confusion matrix and a 2d visualization are shown below.

Matrix 5. Random Forest Confusion Matrix (2 Classes)

[71, 7]
[3, 169]

Figure 11. Random Forest Decision Boundary Binary (2 Classes)



Once again, one sees that the plot and matrix look similar to what would be expected from merging the two coilable classes.

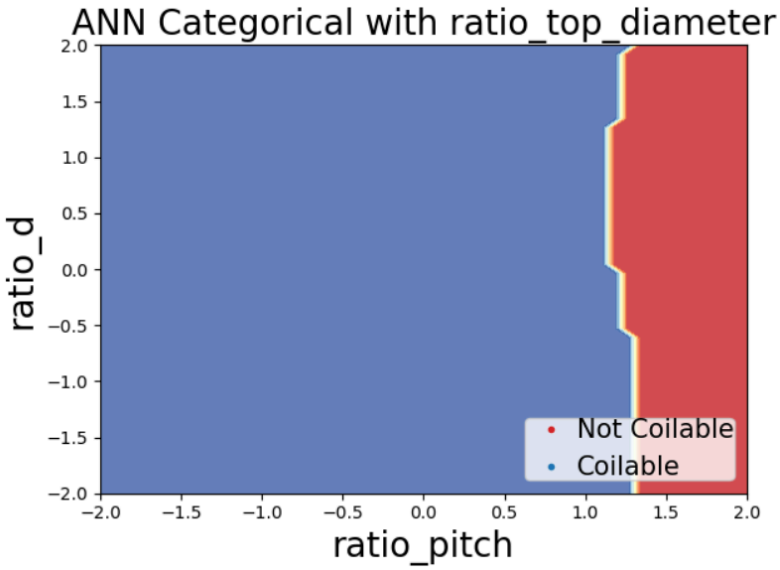
3. Neural Network Classifier

For binary classification, the same Neural Network Classifier had an accuracy of 93.2%. The confusion matrix and a 2d visualization are shown below.

Matrix 6. Neural Network Confusion Matrix (2 Classes)

[72, 6]
[6, 166]

Figure 12. Neural Network Decision Boundary Binary (2 Classes)



The similarity pattern between the 2-3 class decision boundaries holds for the NN as well.

4. Overall Takeaways

Overall, one can see how binary classification is a much easier task than 3-class. For this experiment, there is a lot of physical similarity between structures that are reversibly and irreversibly coilable. This was first seen in the confusion matrices as the values of indices [1,2] and [2,1] were typically 2-3 times the indices [0, 1], [1,0], [0,2], [2,0] indicating that coilable points were more likely to be mistaken for other coilable points than they were noncoilable. In practice, it would be important to consider whether or not yielding is enough of a priority to justify the lower accuracy rate of material classification. For reference, the summary metrics are shown below in Table 8.

Table 8. 3-d Classification Accuracy Metrics (2 Classes)

	Accuracy	Precision	Recall	F1
SVM (RBF Kernel)	0.952	0.956	0.978	0.965
Random Forest	0.960	0.963	0.969	0.967
ANN (two hidden layers)	0.952	0.955	0.978	0.965

For the binary classification, the model with the highest accuracy is now stand-alone at random forest. However, given the nature of the problem the group may still choose to use SVM as the

"best" model for this situation. The slight increase in accuracy may not be worth the increased complexity, especially since this is a problem where the group is relatively familiar with the underlying mechanics. Since it is close, this decision would be at the discretion of the researcher and with this high of an accuracy there really isn't a bad choice among the three models,

iii. Regression

For the regression task, the primary model given in the project instructions is **Gaussian Process Regression**. Known for its sturdy Bayesian roots and strong performance on both small datasets and highly stochastic data, GPR is a computationally heavy regression method. To continue the comparison from the classification task, the two additional algorithms will be **Ridge Regression** and **Artificial Neural Networks**. For this task, Ridge Regression serves as a simpler option to GPR while the ANN serves as another nonlinear model, but as the deterministic option in case of unscalable computation times.

The regression models will be judged by their respective R-squared values and Mean Squared Errors. Note that, for the purpose of comparison, all MSE metrics will be reported in scaled values representing a single standard deviation within the dataset.

1. Gaussian Process Regression

For this task, the GPR performed extremely well on three dimensional data. Yielding a near perfect R-Squared value of 0.999328 and a mean squared error (MSE) of 0.000624. This regression was done on default GPR parameters for a Matern Kernel, using a nu value of 2.5 and a length scale found using the bfgs optimizer with 20 restarts initialized at 1 bounded by [0.01, 100],

2. Ridge Regression

For this task, ridge regression also performed extremely well, with an R-Squared value is 0.999332 and the MSE is 0.000621. Hyperparameter tuning was used to decide on an optimal degree of 5 for the polynomial basis function under the default regularization; this ends up yielding a very similar result to GPR.

3. Neural Network

The neural network still performs well for this task, though it is worse than the results from GPR and Ridge Regression. The R-Squared value comes out to 0.998623 and the MSE is significantly higher at 0.330936. Unlike the prior regression models, the hyperparameters for the Neural Network were decided in an unscientific manner due to the complexity of the model. The final

model was run with 64 neurons on the first hidden layer, 32 on the second, a ReLU activation function (with a linear on the output layer),, a batch size of 200, 1000 epochs, and an Adam optimizer with a learning rate of 0.001.

4. Overall Takeaways

Overall, these models have some clear differences when it comes to regression methods (see Table 9 for the full accuracy metric). By performance, the ridge regression and GPR models perform almost perfectly. The fact that they have very similar performance metrics is not a coincidence. Since the chosen degree for ridge regression (5) is relatively high, the predictions approach those of the Matern GPR which approximate a basis function of infinite polynomials as the data tends to infinity. While one may not see the trend for ridge regression at degrees 1 or 2, 5 appears to be high enough to see conspicuously similar results.

Another trend to note is the difference between these two models with the complex, deterministic Artificial Neural Networks. While commonly seen as the forefront of machine learning, ANNs (especially complex ones) often underperform on smaller datasets as they don't take advantage of the Bayesian framework as much as Ridge Regression and GPR do. While there are additional tactics such as hyperparameter adjustments and regularization that could enhance the performance of this Neural Network, in practice it wouldn't make sense to spend the effort developing those out when either GPR or Ridge already delivers fantastic predictions. At least on the 3D dataset, It is best to adhere to Razor's Occam and use the simplest possible model that solves the problem.

Table 9. 3-d Regression Accuracy Metrics

	R Squared	Mean Squared Error (MSE)
GPR (Matern)	0.999328	0.000624
Ridge Regression (Degree 5)	0.999332	0.000621
ANN (two hidden layers)	0.998623	0.001314

b. 7-d Dataset

To model the 7 Dimensional dataset, the same process was repeated with the provided algorithms (C-Support Vector Classification, Gaussian Process Regression) and the two algorithms self-selected to analyze: Ridge Regression and Artificial Neural Networks. Due to the difficulty

of visualizing results on 7 dimensions, these results are communicated using the summary tables below, followed by a brief summary of the overall takeaways.

i. Classification

In summary, the accuracy of these three methods for 2-class classification is shown in Table 10.

Table 10. 7-d Classification Accuracy Metrics (2 Classes)

	Accuracy	Precision	Recall	F1
SVM (RBF Kernel)	0.958	0.957	0.951	0.954
Random Forest	0.952	0.951	0.943	0.947
ANN (two hidden layers)	0.970	0.969	0.988	0.978

For this classification task, the Artificial Neural Network now has the best performance. Note that all three models were still optimized using the same methods: grid search for SVM yielding C=100, gamma=0.7, grid search for Random Forest (yielding n_estimators=200, max_depth=None, min_samples_split=5, min_samples_leaf=1); and the same NN architecture and hyperparameters from the first classification, with the exception of and increased number of epochs (1000) to account for additional features.

ii. Regression

In summary, the accuracy of these three methods for regression is shown in Table 11. Note that the MSE values are scaled quantities.

Table 11. 7-d Regression Accuracy Metrics

	R Squared	Mean Squared Error (MSE)
GPR (Matern)	0.964031	0.032061
Ridge Regression (Degree 7)	0.978428	0.017931
ANN (two hidden layers)	0.984236	0.014808

The artificial neural network also has the best performance for the regression task. Note that all three models were still optimized using the same methods as on the 3d dataset: grid search for GPR under the same constraints as the 3d data, grid search for ridge regression now yielding a

7th degree model, and the same NN architecture and hyperparameters from the first regression, with the exception of and increased number of epochs (1000) to account for additional features.

c. Comparing Models: Dimensionality

The above analysis provides interesting insights on the use of each algorithm as it pertains to dimensionality and the amount of available data. Among both classification and regression there was a clear flip in performance. For smaller datasets with less input features, the models with stronger bayesian theory outperformed their deterministic counterparts. For 3D, both the C-Support Vector (RBF kernel) and Gaussian Process Regression (Matern) displayed very high success rates even with a relatively small dataset, while the Neural Network struggled to keep up at that level of data. This reveals a key insight on machine learning's role in mechanics: **since datasets are normally small** due to the immense computation required for engineering simulation, many problems will be **better suited to strong Bayesian models**.

However, it is worth noting that Neural Networks and more deterministic models can have success on models with more data and input features, as shown by the 7-dimensional datasets. As FEA simulations become more efficient and advanced, it is possible that these complex, deterministic algorithms will play an increased role in metamaterial design.

d. Comparing Models: Classification vs. Regression

Another noteworthy difference in these models' behaviors is the differences in success by task as the dimensions increase. For every model, while adding dimensions and data made the classification task easier it made the regression task more difficult. This remains true when directly comparing the two class classification between the two datasets. This result, while seemingly contradictory, represents the physical nature of the problem. Adding more tunable variables to model design provides more information about what facilitates coilability without changing the possible outputs (it remains just coilable or not coilable). However, adding these tunable variables affects the underlying mechanics of the problem which introduces new ways for the continuous variables (sigma critical, energy) to behave. For example, a certain combination of features 6 and 7 could make possible a new combination of stress and energy that wasn't possible with the three original features.

e. Scalability

Scalability to problems even larger than 7 Dimensions is a key component to this work, both in model quality and in required computation. GPR is the model on which scalability is the biggest issue when considering these two components. Those stand-out results from the low data / low dimension task do not continue to scale with this 7d problem, as the other regression forms catch

up in terms of accuracy. Additionally, the runtime balloons to an order of magnitude higher than either of the other four algorithms that were explored throughout the course of this project (this includes two that didn't make it into the final paper).

On the other hand is Neural Networks. While the lack of Bayesian strictness hurts them on the smaller, more straightforward dataset; this same deterministic nature facilitates shortcuts that are very helpful in the complex 7-dimensional feature spaces (and above). Combining that predictive power with an advanced runtime due to point estimation gives a very promising model for larger problems.

3. Comparing Models

a. Influence of Training Points

An important factor in how well a machine learning model performs is the amount of data provided for it. In this experiment, the impact of data quantity can be explored by varying the test train ratio in the 1000 point 3-dimensional data set. The plots below show the training ratio changing from 5% to 95%, hence the training points varying from 50 to 950. Note that since the training ratio is 1 minus the test ratio, the accuracy values (measured on test data) may experience random noise as test ratios approach zero.

Figure 13. SVM 3-d Classification Performance by Test Ratio (1000 total points)

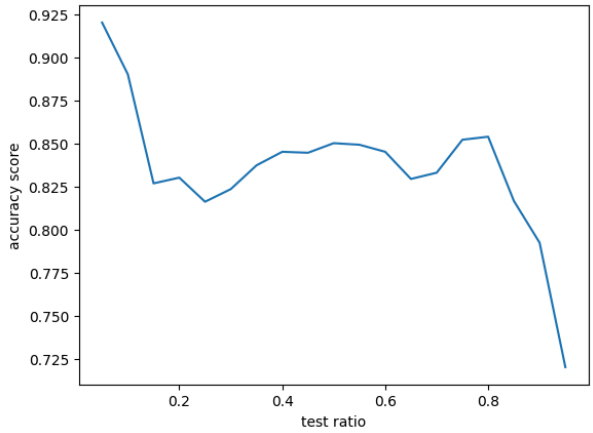
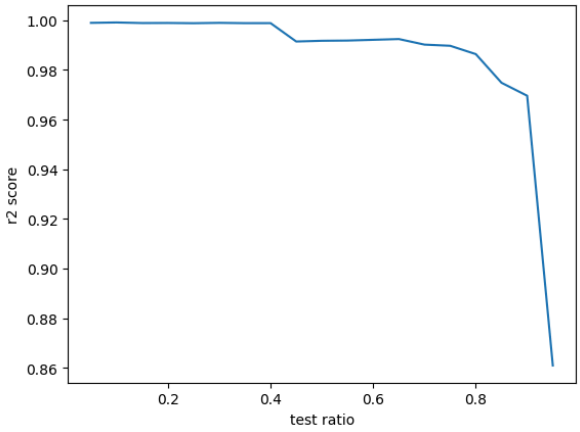


Figure 14. GPR 3-d Regression Performance by Test Ratio (1000 total points)



For both cases, it is expected that as the number of training points increase, the accuracy increases. While the data from the above figures supports this trend in general, it is not always monotonically true. The classification, for example, has multiple peaks and no clear trend outside of the two extremes (very low training data and very low testing data). The extremely high classification results at low test data (>90%) are likely due to random noise in the test data, as

there wouldn't be such an extreme jump in going from 850 to 950 training points. Further investigation into this phenomena is possible but outside of the scope of this project.

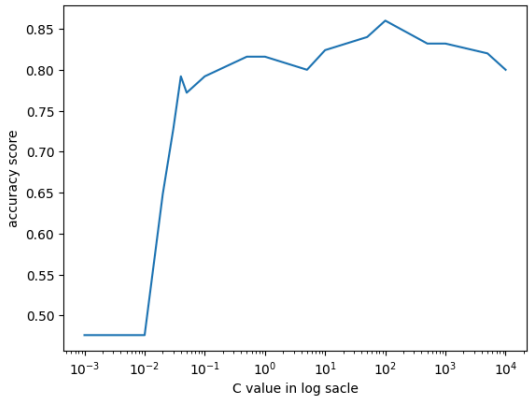
Overall, the group feels comfortable with the standard test ratio of **25%** for this problem. It seems to get the optimal results for regression. And there is technically a higher accuracy at higher test ratios for the classification, changing that model hyperparameter just because we've seen it has higher accuracy would be bad practice without any other logic supporting it (and it could potentially lead to overfitting).

b. Influence of Hyperparameters

i. C Support Vector Classifier

For the C Support Vector Classifier, the C parameter plays a significant role as a regularization parameter. See Figure 15 below to see the output accuracy ranging C from 10^{-3} to 10^{4}

Figure 15. SVM 3-d Classification Performance by C.

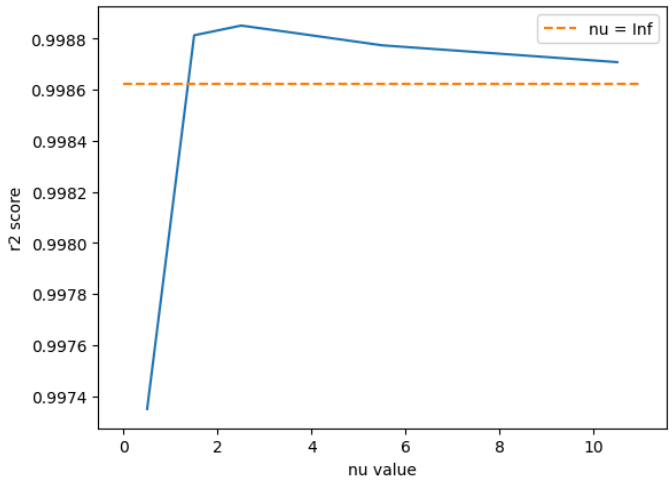


From Figure 15, it appears that the optimal **C value peaks around 100**, resulting in ~85% accuracy. This later peak with additional regularization came as a surprise to a group. In fact, this exercise caused a revision of part 2 with adjusted bounds for the grid search for hyperparameter selection, resulting in a 4% increase in accuracy. This event highlighted the importance of hyperparameter optimization for classification problems, and helped guide the group's tuning of hyperparameters of all models used for the remainder of the project.

ii. Gaussian Process Regression

For the Gaussian Process Regression, the smoothness parameter (nu) of the Matern kernel plays a significant role as it controls the smoothness of the kernel function with higher values leading to heavily smoothed functions and lower values resulting in rougher, more complex data patterns.Figure 16 shows GPR performance as the value of nu is varied from 1 to 11.

Figure 16. GPR 3-d Regression Performance by Smoothness Parameter (Nu)



From Figure 16, the result of the R-squared score, increasing before nu = 2.5 and decreasing after it, reveals the fact that for this problem particularly, **nu = 2.5** is most likely the smoothness of the target function. While two ideal cases, nu = 0.5 and nu goes to inf, will all give less accurate results compared to nu = 2.5. According to the document of the Matern kernel, nu = 0.5 will lead to absolute exponential kernel, and nu going to infinity will lead to radial basis function (RBF) kernel [3, 4, 5]. In practical cases, nu is between these two ideal cases and is dependent on problems.

4. Optimization

a. Preprocess of optimization

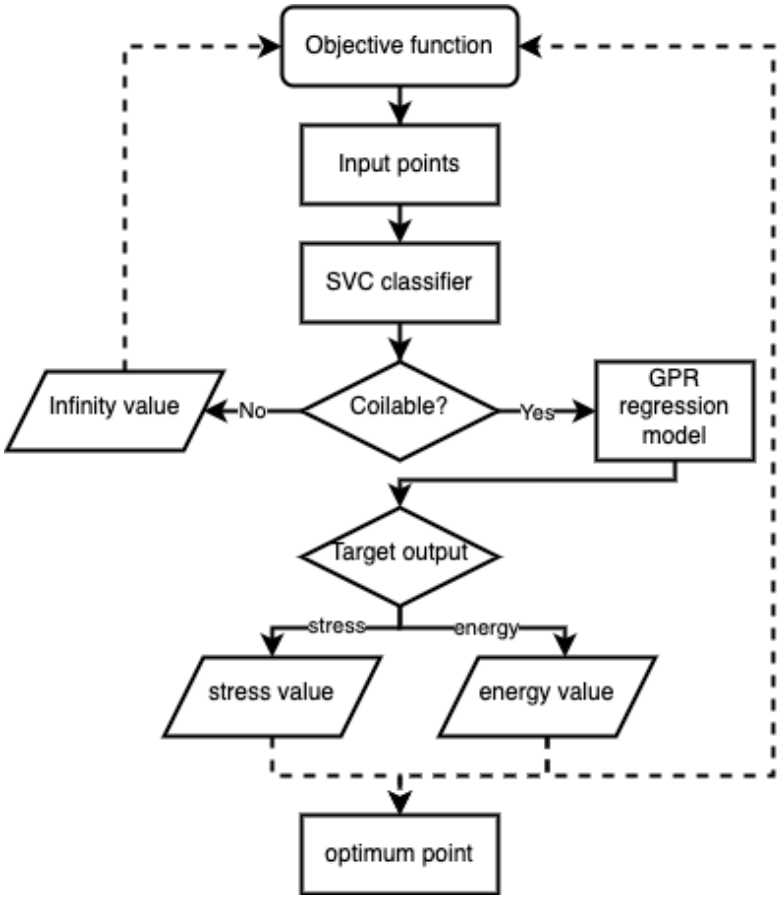
With a well fitted classification model, C-Support Vector classifier, points with highest critical stress and absorbed energy can be searched directly in a 3-dimensional input and 2-dimensional output dataset. Following this process, a point with highest critical stress, as 4.969 KPa, is found as (0.032, 0.449, 0.665), and a point with highest critical stress, as 3.732 $KJ \cdot m^{-3}$, is found as (0.034, 0.321, 0.758). These two points are shown in Table 12. The difference of critical stress between the maximum number in Part 1 and the searched point in this part comes from the strategy to do the preprocess of this dataset, which is dropping those points whose outputs include "NAN" in raw dataset integrating "supercompressible_3d_input.csv" and "supercompressible_3d_output.csv" together. Two searched points with highest stress and energy values will be the initial points in further optimization process, and represent as a reference of optimization effects as well.

Table 12. Optimization results with input points

Algorithms	$\sigma_{crit}/(KPa)$; (r_top_diameter, r_pitch, r_d)	$E_{abs}/(KJ \cdot m^{-3})$; (r_top_diameter, r_pitch, r_d)
Part 1 dataset maximum	84.408; (0.072, 0.976, 0.477)	3.732; (0.034, 0.321, 0.758)
Search Point in Dataset	4.969; (0.032, 0.449, 0.665)	3.732; (0.034, 0.321, 0.758)
Nelder-Mead method	118.431; (0.076, 0.977, 0.526)	47.589; (0.063, 0.686, 0.669)
L-BFGS-B method	9.902; (0.038, 0449, 0.665)	9.231; (0.042, 0.321, 0.758)
Powell method	65.394; (0.057, 0.526, 0.773)	38.898; (0.057, 0.561, 0.760)
TNC method	9.971; (0.038, 0.449, 0.665)	9.277; (0.042, 0.321, 0.759)

Before implementing the optimization process, a clear objective function is needed for all optimization algorithms. Figure 17 shows the return values of the objective function with input points and target value (critical stress or absorbed energy). If the classifier classes the input point as a coilable point, the GPR model will take in this point and make a prediction for target value, e.g. critical stress, and then, the objective function will return this stress value to the optimizer. Considering another case, if the classifier predicts that the input point is not coilable, then the objective function will directly return an infinity value to the optimizer. For particular optimizers, which cannot take in an infinity value, in this case, the objective function will return 100,000 as an approximation of the infinity, for instance, the TNC and L-BFGS-B methods need this approximation in practice.

Figure 17. Flowchart of the objective function



b. Different optimization algorithms

Nelder-Mead is the firstly tried method in the optimization process. Not only because it is a mandatory requirement for the project but also because it does not require the gradient of the objective function and line search [6], which means it is simple to implement. It should always be the first try for small-dimensional problems and non-smooth optimization problems. Obviously, according to Table 12, the Nelder-Mead method gives wonderful optimus of both critical stress and absorbed energy, increasing the critical stress to **118.431 KPa** at (0.032, 0.449, 0.665) and the absorbed energy to **47.589** $KJ \cdot m^{-3}$ at (0.063, 0.686, 0.669).

The **L-BFGS-B** method is known as a powerful and efficient algorithm and well used in the course lecture. So optimization with it in this problem may help for further understanding of its benefits and shortcomings. However, the L-BFGS-B method does not give optimal values as high as the Nelder-Mead method, and only results in **9.902 KPa** for critical stress at (0.038, 0449, 0.665), and **9.231** $KJ \cdot m^{-3}$ for absorbed energy at (0.042, 0.321, 0.758). Proposed reasons for this low value optimums are misguiding by the gradient of object function and ineffective line search. These two problems might both be due to the noisy object function this optimizer is

trying to deal with. If the objective function is noisy, the gradient computed at each iteration may be inaccurate or misleading. This can cause the algorithm to make poor decisions about the direction in which to move. Meanwhile, noise can cause fluctuations in the function values, making the line search might fail to converge, or it might converge to a suboptimal step size.

It is interesting that those fancy techniques like gradient descent and line search might play negative roles in this particular problem. Also, it will be interesting to check which one of these two techniques might dominate the poor performance of the L-BFGS-B method. The TNC method, which only uses gradients for searching directions and the Powell method, which only uses line search strategy, are implemented for further exploration of different optimizers.

The **Powell** method, which is a derivative-free optimization algorithm that seeks the minimum of a function by maintaining a set of conjugate directions with line search technique. The results as **65.394 KPa** at point (0.057, 0.526, 0.773), and **38.898** $KJ \cdot m^{-3}$ at point (0.057, 0.561, 0.760) show better optimums than the L-BFGS-B method, but worse results compared to the basic Nelder-Mead method. It means, on the one hand, removing gradient search might improve the performance of the optimizer for the supercompressible problem, on the other hand, line search is not a suitable technique for this problem either.

Finally, the **TNC** method, which uses gradients for searching but does not use the line search technique, is applied. It give similar results as the L-BFGS-B method, that the critical stress is **9.971 KPa** at (0.038, 0.449, 0.665) and the absorbed energy is **9.277** $KJ \cdot m^{-3}$ at (0.042, 0.321, 0.759). The almost same results with the TNC and the L-BFGS-B methods illustrate that this low value optimum might be caused by the gradient based search strategy.

Figure 18 and 19 show the convergence of optimization processes for critical stresses and the absorbed energy. Comparing the interaction numbers when each method achieves the optimums of critical stresses, L-BFGS-B and TNC methods converge firstly within 20 times searches when they almost find the optimum. Then, the Powell method gives a nearly optimum with around 40 times searches. Finally, the Nelder-Mead method converges with 80 times searches.This series is reasonable according to documents of those methods, for instance, the Nelder-Mead method does not use gradient descent and line search, which tends to cause problems in time efficiency. Moreover, when zooming into the platform stages of each method, the Powell shows a noticeable vibration which means that it is still trying to search in different directions. During these searches, the Powell method gets some worse results and then just comes back to the optimum which it has already got.

Figure 18. Convergence curve of stress optimization with three methods

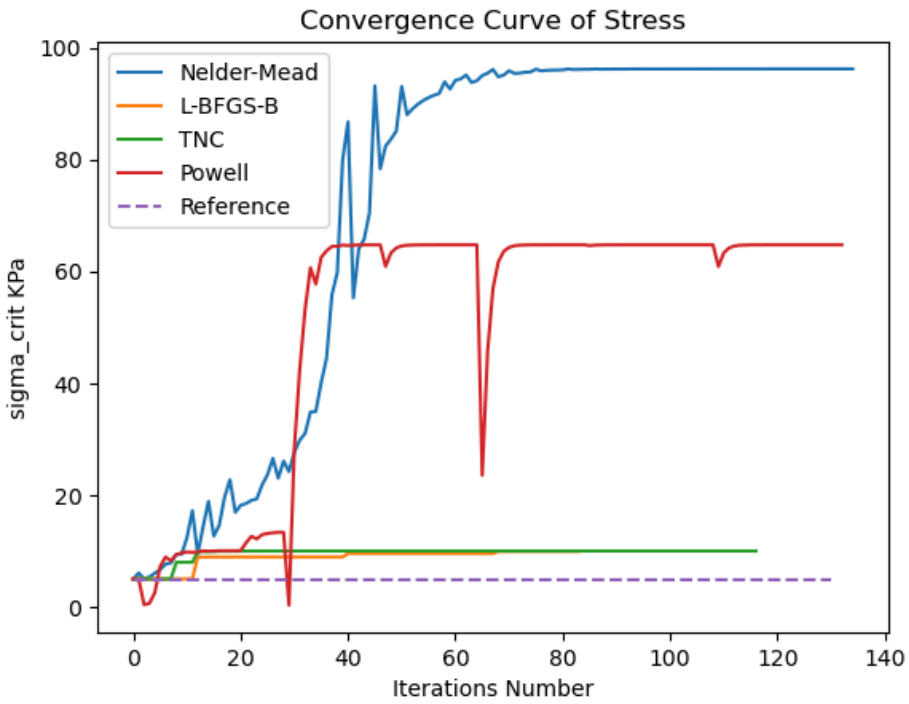
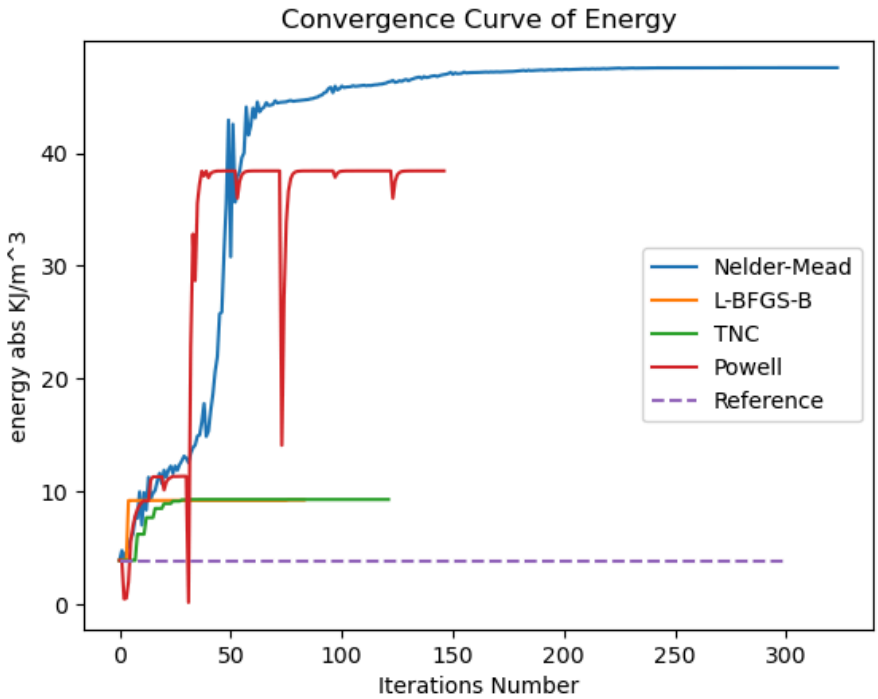


Figure 19. Convergence curve of energy optimization with three methods



5. Conclusion

Overall, the results in the above analysis show the many variables involved in model and hyperparameter selection for machine learning in engineering. They also show the amount of trial and error that goes into developing these models (for both the machine learning and optimization objectives of the project). The group's biggest takeaway from this analysis is that there is often no single answer when it comes to machine learning, even for the same underlying problem! Depending on the data count, dimensionality, and personal preferences there are so many nuanced ways to develop and choose models that could work for this application.

6. Bibliography

[1] M.A. Bessa, P. Glowacki, and M. Houlder. Bayesian machine learning in metamaterial design: Fragile becomes supercompressible. Adv. Mater., 0(0):1904845–, October 2019.

[2] M. A. Bessa, R. Bostanabad, Z. Liu, A. Hu, Daniel W. Apley, C. Brinson, W. Chen, and Wing Kam Liu. A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality. Computer Methods in Applied Mechanics and Engineering, 320(April):633–667, jun 2017.

[3] Pedregosa et al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011.: sklearn.gaussian_process.kernels.Matern

[4] Carl E. Rasmussen and Christopher K.I. Williams, Gaussian Processes for Machine Learning, MIT Press 2006, Chapter 4

[5] David Duvenaud. The Kernel Cookbook: Advice on Covariance functions, 2014

[6] Kevin P. Murphy, Probabilistic Machine Learning: An introduction, MIT Press, 2022. Chapter 1: 265–300