Developing a machine-learning model to predict perovskite film lifespans using experimental data

Aryan Johari ajohari@ucsd.edu
Chi Wong csw002@ucsd.edu
Lucas Papaioannou lpapaioa@ucsd.edu
Jacob Kell jkell@ucsd.edu
David Fenning dfenning@ucsd.edu

Abstract

As it currently stands in order to know the lifetime of a perovskite film we must wait weeks or even months until the film is fully degraded. This is a problem as we will have to wait weeks or months to know for sure if a change in a chemical's molarity will cause the film to degrade faster or slower for example. In this paper, we tackle the problem by making a machine learning model that provides insight on the lifetime of a perovskite film at an early stage in the process. With help of the model, researchers would be able to accelerate the experiment procedure and gain early insight regarding the impact of a change in experiment on the lifetime of the film.

Code: https://github.com/ajohari114/dsc180-solar2

1 Introduction

Perovskite cells are hailed to be the next generation of photovoltaic cells that can theoretically be more efficient than traditional silicon cells. One of the weaknesses of perovskite cells is that they tend to degrade at a much faster rate under direct sunlight than silicon cells. Researchers around the world are committed to decipher the factors contributing to their rapid degradation and explore strategies for optimizing their production to address this issue.

Dr. Fenning's lab employs an experimental method involving various samples of perovskite film using different synthesis conditions and testing them under accelerated conditions – intense sunlight and high temperatures – to track degradation speed of these films. The challenge is that these experiments can last for weeks to months before conclusions are clear regarding the synthesis conditions influencing degradation and how to design meaningful future experiments. A potential solution is to train a machine-learning model on past experiments so we can learn which factors significantly impact film degradation so we can optimize future cell production for longer durability.

This is inspired from research done by Hartono et al. (2020) where they utilize machine learning models on accelerated aging tests on perovskite layers and determine the important features that impact degradation through feature ranking Hartono et al. (2020). While our approach is going to be similar, what we hope to change is to build a robust continuous learning system which handles data flow from the engineering side such as data ingesting, preprocessing and database managing, to machine learning work like model fitting, training and operating. The continuous learning model provides agility to continuously adapt the rapidly adjusting experiment data from the lab and improve its performance over time while ensuring data freshness. Not only would such a model provide an ML prediction for degradation of solar cells before manufacturing, it also provides a potential data-driven direction for our ongoing research. The goal for the project is to provide an insight at an early stage in the experiment, effectively reducing the idle time down to the unit of days, or even hours.

2 Data Description

The data we are using comes from Dr. Fenning's lab which involves how cells of differing chemical compositions, manufacturing steps, or solvents react to highly intense sunlight and high temperatures. The data is stored in a graph database where each node has different characteristics about an experiment or features but has parent attributes such as sample_id, batch_id and step_id, each referring to the experiment sample, batch of samples, and the step in the experiment respectively. Further, we also possess images of sample perovskite films from which we can extract colormetric data to further understand degradation behavior. To ensure data freshness and optimize data for model consumption, we constructed a data pipeline that enabled the seamless flow of data from the lab's hardware machine, which makes the perovskite film to a Synology storage drive, with daily

synchronization to our production graph database. The pipeline encompassed data collection, transformation, and storage components, leveraging robust data handling techniques and technologies. The functionality to synchronize data from our Synology drive to the graph database is implemented by setting up a Linux cron scheduled job. The daily running job copies new data in the drive to a new path to provide extra security protection to the valuable lab data, then executes a script to write the new data into the graph database instance. Our data preparation approach centers on maintaining data completeness and enhancing existing data in the database. This is achieved by establishing a local Neo4j graph database,with nodes and relationships with distinct properties such as batch_id and sample_id, mirrored from characteristics of the production data. The code simulates a graph database schema in which data nodes are interconnected based on their unique batch, sample, and step ID identifiers. Currently our dataset in the Neo4j graph database is limited and doesn't have quite enough samples to do effective machine learning but more samples are being added to the graph database that will enhance the dataset. As we continue to expand our graph database with additional samples and more experimental data, we anticipate that we will be able to further leverage machine learning for more modeling and data-driven insights in regards to stability and performance of perovskite-based solar cells.

3 Methods

3.1 Image processing

This section of the image analysis was already done by the Fenning research group prior to us joining them. The process follows processing a series of images to measure the color change of cells from black to yellow, yellow meaning completely degraded. While this process was already done by the research group, we optimized a few things.

3.1.1 Template Matching

We employed OpenCV's template matching tem (2199). The way it works is we have a precropped region of an image with all the sample cells – we call this the baseline image. The algorithm loops through the entire series of images and finds the most similar region of each image to the baseline and crops that region. Although this may seem inefficient, this is essential as a fail safe incase the camera moves in the experiment or changes angle.

3.1.2 Improving Space Complexity

The previous iteration of the code was quite memory intensive as it loaded all the images into memory by making a video matrix of all the samples and performing the analysis at once. We solved this problem by loading only a portion of the images at a time and integrating the color analysis while the original code iterated through the samples to create

a video matrix. This meant our code's space complexity was $\theta(n/k)$, where k is a segmented portion of the data instead of $\theta(n)$ with respect to the number of samples in an image.

3.1.3 Colormetrics Parameter Fitting

This is a crucial step in this project as it produces the parameters that are used to approximate the degradation trend of each film.

This involves first using a Gaussian function to smooth out the degradation trend line to get rid of noise. Then, a Logistic function is fit to the curve, outputting three parameters: `L` which is the point at which the curve plateaus, `x0` which is number of hours where the sample has half degraded, and `k` which is rate of degradation. Then, the curve is normalized, which results in `L` always equating to 1, which returns normalized values of `k` and `x0`.

A visual example of this process is shown in Figure 1 below.

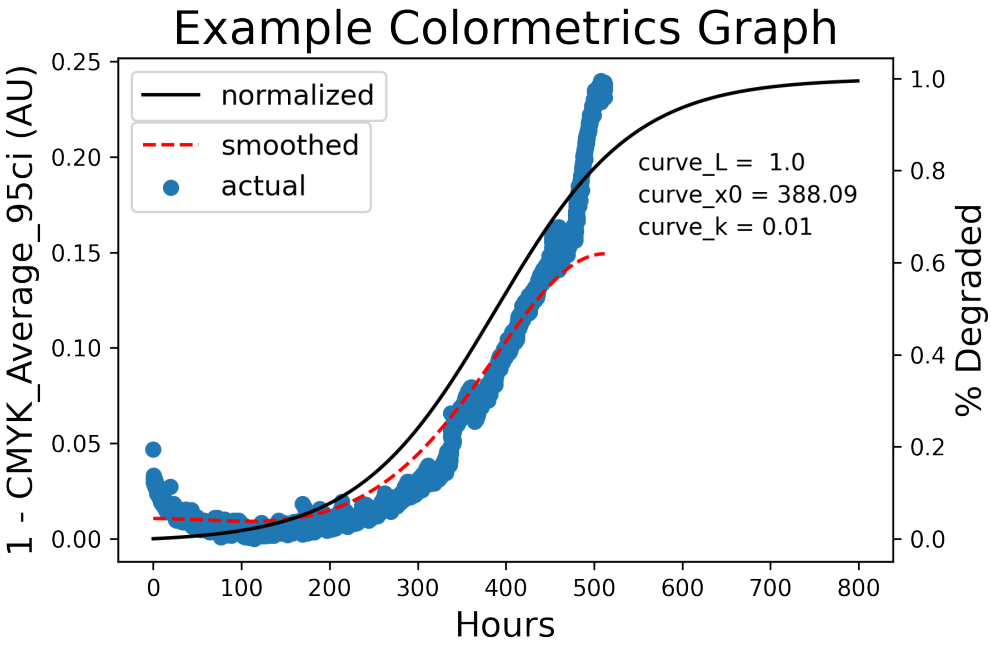

Figure 1: Example of fitting a logistic curve to the colormetrics.

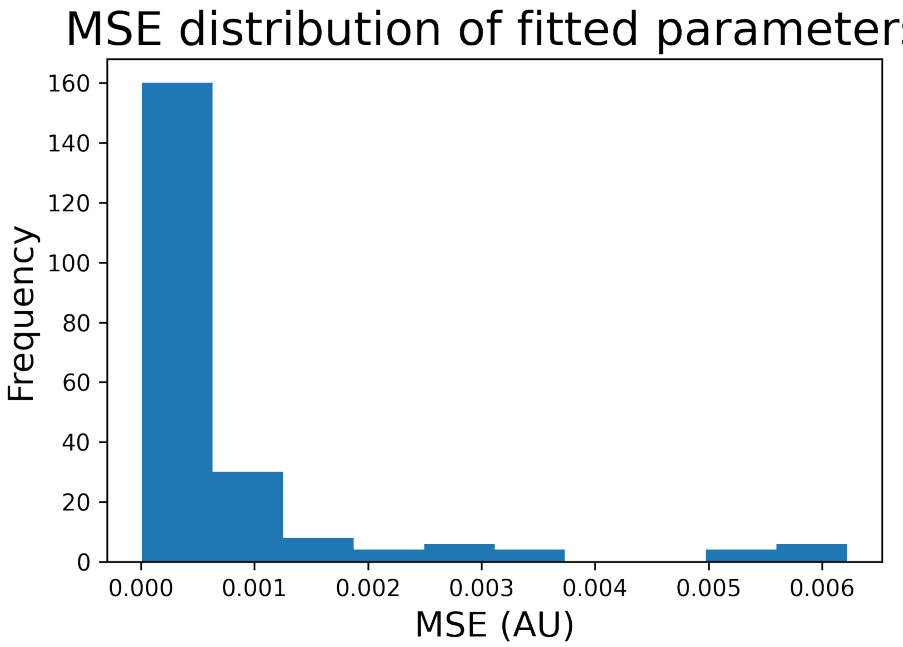

Figure 2: MSE distribution of the logistic curves and the true colormetrics curve

This fitting procedure was quite successful as Figure 2 indicates that the average MSE is quite low for a sample size of 147 total films. This means that one can successfuly use the fitted logistic curve as an approximation for the degradation trend of a perovskite film.

3.2 Graph Database Pipeline

For this pipeline, we developed a script that uses the Py2neo package to interact with a Neo4j graph database hosted on a Network Attached Storage that stores information describing the properties and manufacture processes of experimental solar films. The function `grab_sample(batch_id, sample_id)` uses a Cypher query to retrieve all the `Action` and `Chemical` nodes associated with the passed `batch_id` and `sample_id`.

The `grab_batch(batch_id)` function uses a Cypher query to retrieve all the samples that are associated with the passed `batch_id`. A list of `batch_id` and `sample_id` pairs are returned that would be used in the `tabularize_samples` function.

The `node_cluster_size(batch_id, sample_id)` function uses a Cypher query to retrieve all the nodes associated with the passed `batch_id` and `sample_id` and returns the number of nodes that were retrieved.

The `all_samples()` function returns a list of `batch_id` and `sample_id` pairs of all the samples within the Graph Database.

The `segment_samples(rules)` function returns a list of `batch_id` and `sample_id` pairs of the samples that pass the list of conditions that are passed to this function. If the cluster size of each sample that is filtered is not the same a warning is thrown saying that

tabularizing this list of samples is not recommended.

The function `tabularize_samples(samples)` takes in a list of `batch_id` and `sample_id` pairs and returns a DataFrame where each row is a sample and the columns describe the properties and manufactory processes of that sample. A machine learning model would then be able to be trained on the tabularized data.

The function `create_row(sample)` is used by the function `tabularize_samples` to create rows for a data frame. It takes in a data frame representation of a single sample where each row is a node for that sample. The nodes of the sample are then iterated over and relevant information is extracted to create a data frame row.

3.3 Machine Learning Pipeline

The machine learning pipeline takes the data from the graph database pipeline and separates it into features that are used for prediction and the curve parameters which are the target variables or the values being predicted by the model. Our approach was to train a model separately for each x0 and k parameter as training them both in a multiregressor showed worse results. Some of the features were either strings or boolean values so they were one-hot encoded in order to be able to fit the model. We did research on what the best models are for data poor environments and decided to use these models which included Ridge Regression, Lasso Regression, Elastic Net, Support Vector Regression(SVR), Random Forest, and Catboost. We also tested hyper parameters using both gridsearch, which tests multiple parameters and figures out which parameter is best for each model, and cross validation, which splits the data into 80% train and 20% test data based on different folds. For hyperparameters we tested alpha for Ridge Regression, Lasso Regression, and Elastic Net, and l1_ratio and max iterations as well for Elastic Net. For Random Forest Regressor we tested varying amounts of max depth of the tree, the minimum amount of samples it is split into, the amount of estimators, and the max features. For SVR, we tested types of kernel, and the degree and C values. For our Catboost model we did not test any hyperparameters. We also tested different amounts of folds to see which ones would be best.

3.4 Database Operation Pipeline

The Database Operating Pipeline in our project is specifically designed to streamline the process of fetching data from a Network Attached Storage (NAS) system and updating our graph database on a daily basis. This component is centered around a script that automates the extraction of relevant information from the stored data and its subsequent integration into the graph database. This ensures that the database remains up-to-date with the latest experimental data. By automating this process, we significantly reduce the manual effort required for database maintenance and improve the efficiency of data management. This pipeline is integral to maintaining the freshness of our dataset, which is critical for the accuracy and reliability of the machine learning models. The daily update routine is designed to be robust and secure, thus supporting the continuous learning system in adapting to

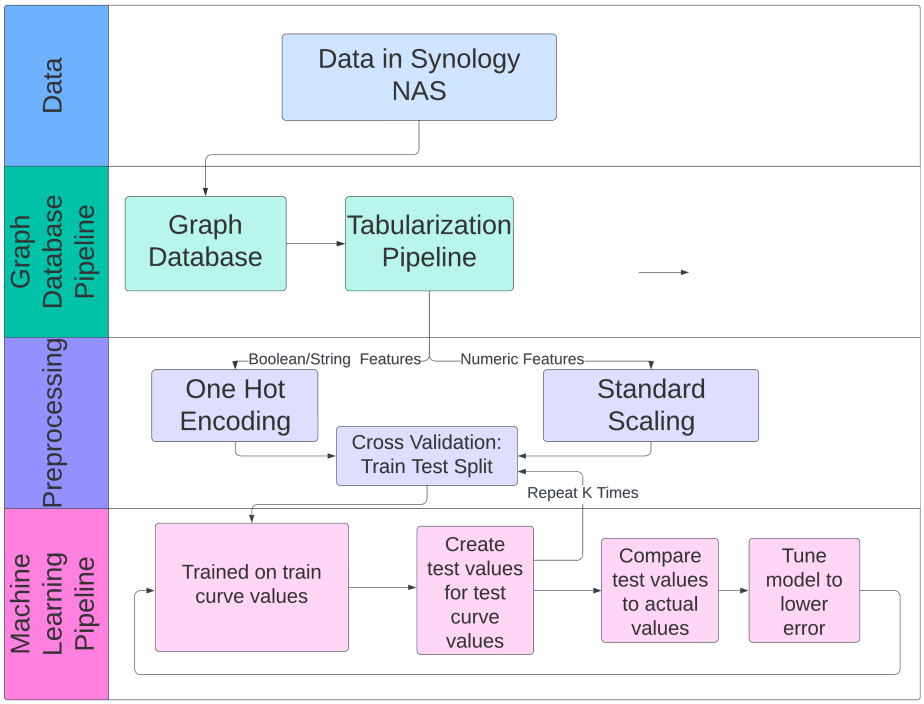

Figure 3: Flow chart depicting entire process from Synology Data to Final Model

changing experimental data.

4 Results & Discussion

We evaluated the best model based off the model that produced the best cross validation score. We had 2 target variables to predict for the curve parameters which were curve_k and curve_x0. We also tested whether building models for each curve parameter or building a multiregressor that includes both target variables would produce a better result and found that creating models for each of the target parameters separately creates a better model. For predicting both curve_x0 and curve_k, we found that CatBoost produced the best model in terms of minimizing the raw RMSE.

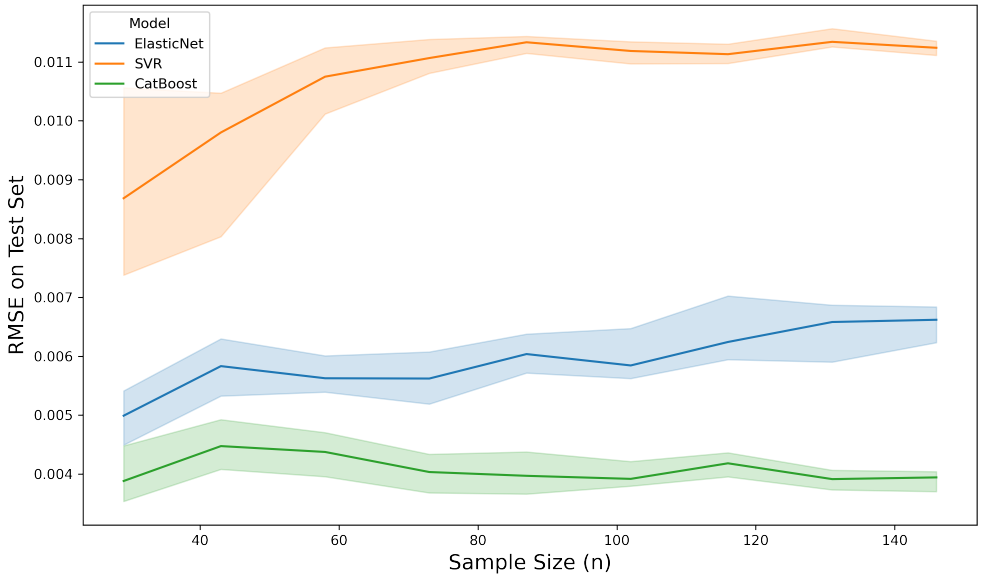

Figure 4: Model performances vs training sample size in predicting k (100 trials).

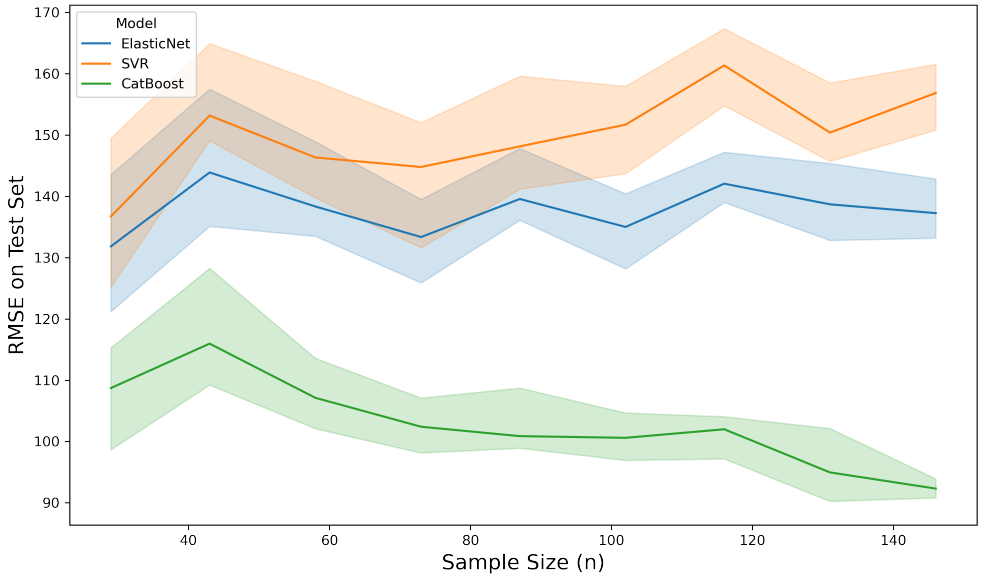

Figure 5: Model performances vs training sample size predicting x0 (100 trials).

The purpose behind producing Figures 4 and 5 is to study how our models perform at different training sample sizes. This is important to evaluate as the number of samples in a full data set is likely to be small for the foreseeable future. Figure 4 shows that the RMSE for all 3 models are relatively stagnant as the sample size increases. The bands around the trendlines indicate the 95% confidence interval of the RMSEs obtained throughout the 100 trials. The bands decrease in width as the sample size increases, meaning that there is low variance in the results of the models as sample size increases. A similar trend is observed in Figure 5 where the variance for all the models decreases drastically as the sample size increases, however unlike Figure 4, the RMSE for CatBoost decreases noticeably.

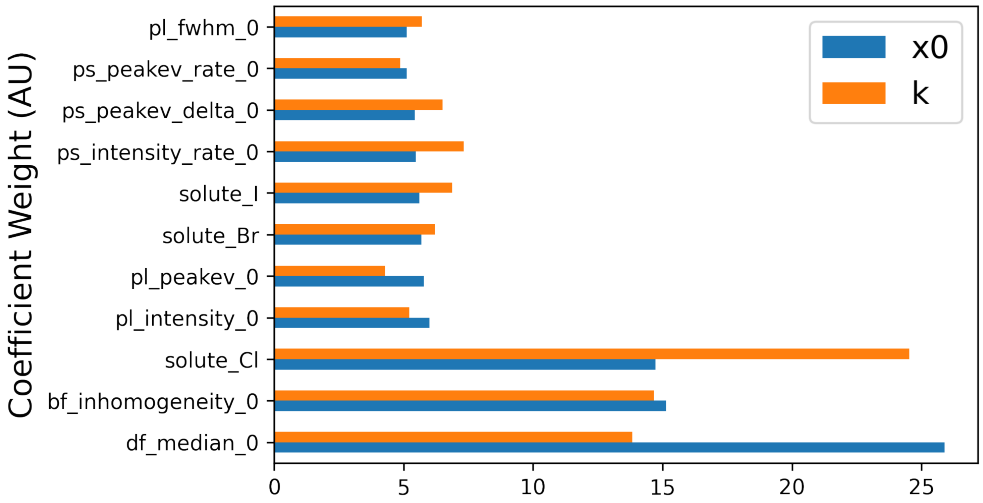

Figure 6: Feature importance extracted from CatBoost

Figure 6 shows the results from the CatBoost model that was trained on the composition features and the physical characteristics of the films. The results indicate that the chlorine solute concentration `solute_Cl` has the greatest influence on the degradation rate of the film `k`, whereas the feature `df_median_0` has the greatest influence on prediction the onset `x0`.

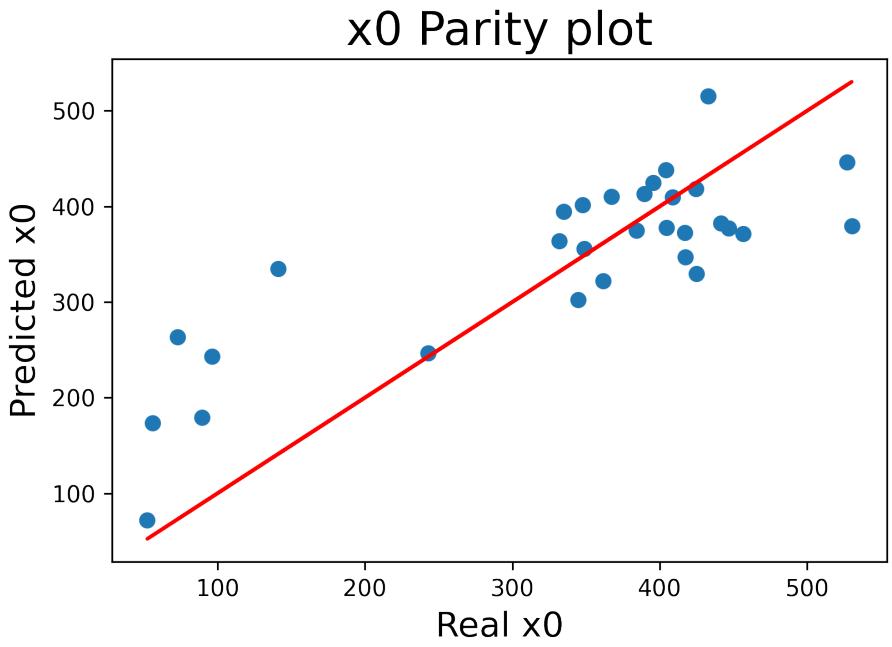

Figure 7: x0 parity plot from one of the better train-test splits (random_state = 2)

Figures 7 and 8 show parity plots from one of the better train-test splits of our full dataset. These charts show that as k grows, the the the predicted k does not grow with it, meaning that the models tend to underpredict the rate of degradation. However, the opposite is true for x0, as the model performs better at higher x0 values.

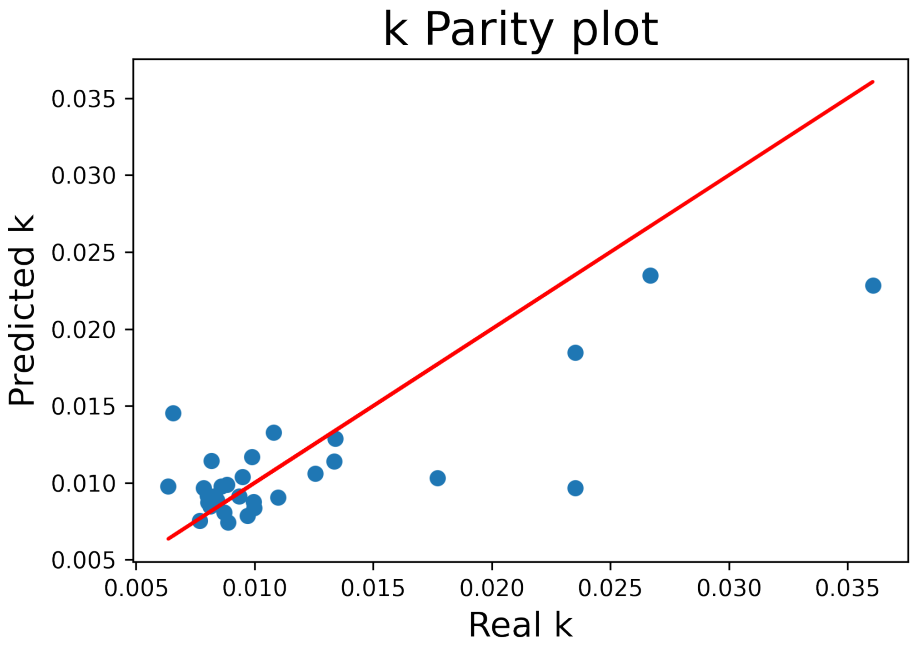

Figure 8: k parity plot from one of the better train-test splits (random_state = 2)

Upon further investigation, we noticed that `x0` and `k`, from the fitted colormetrics parameters, in fact have a dependent relationship. This is shown in Figure 11 below. To explore this further, we decided to predict only `x0` alone and use that value to extrapolate a predicted `k` value.

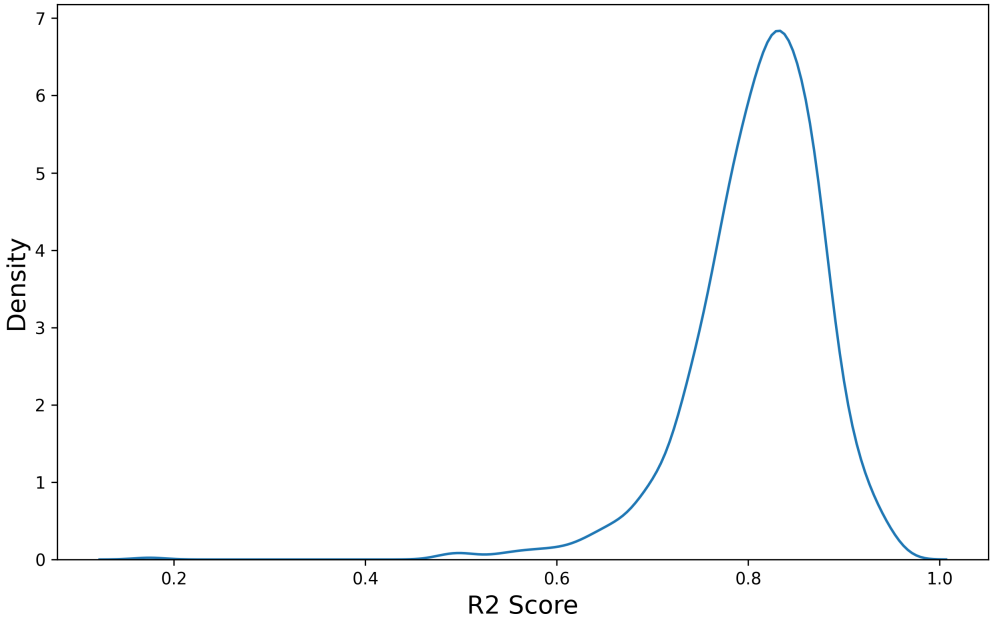

Figure 9: R^2 density graph for k predictions

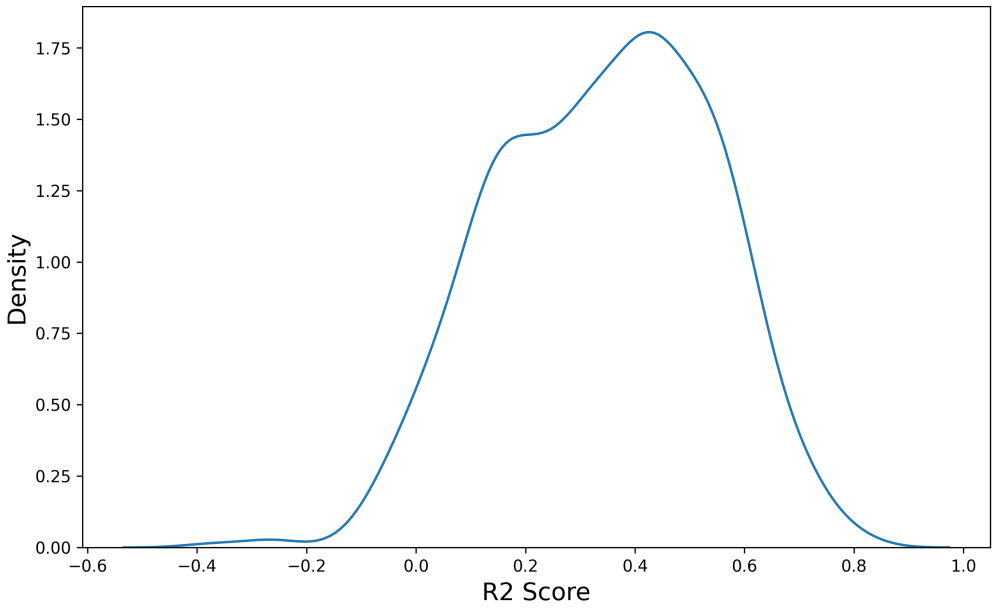

Figure 10: R^2 density graph for x0 predictions

The improved prediction performance in k is shown by Figure 9. As can be seen the model for predicting k generally performs better than the model predicting x0. This does make sense because, as seen in Figure 11, there is a relationship between x0 and k that is similar to $y = \frac{1}{x}$. Theoretically, this is a simpler feature space to predict in comparison to the feature space when predicting x0. This does seem to be supported by Figure 9.

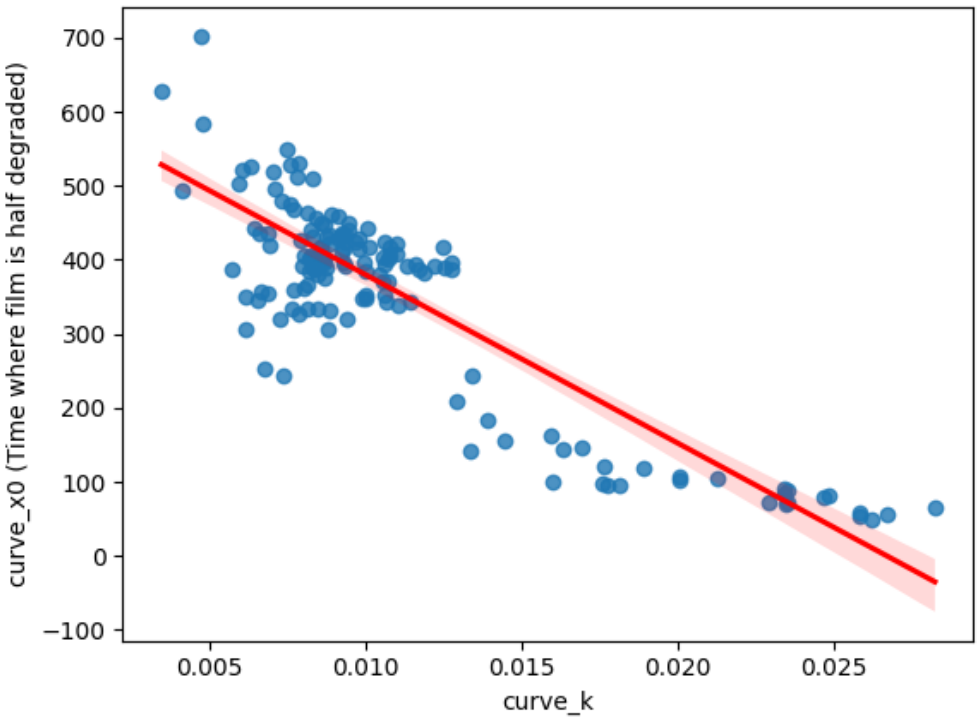

Figure 11: Dependent relationship of x0 vs k derived from fitted parameters

5 Conclusion

Overall, the project yields a promising road for using machine-learning models to aid expedite experimentation for Dr. Fenning's research group. CatBoost is a great ML candidate to predict the lifespan of perovskite films. There is still a large room for improvement, however. One potential area is to further explore models that can predict both x0 and k simultaneously. This was a challenge for us since the models we were testing were not compatible with a multioutput label, hence we resorted to predicting x0 alone and using that to predict k.

Another area of improvement would be to train on certain segments of the data alone, so perhaps samples that correspond to a high rate of degradation or low etc. This could improve model performance by being fit for certain "ranges" of degradation.

Lastly, it would be great to explore using the pattern of degradation on the images as a feature, which could promp convolutional neural networks as a mode to get more features

out of the images. This can be an exciting avenue to explore in the future as there could be unseen correlations between the pattern of degradation and the synthesis of the films.

References

Hartono, Noor Titan Putri, Janak Thapa, Armi Tiihonen, Felipe Oviedo, Clio Batali, Jason J Yoo, Zhe Liu, Ruipeng Li, David Fuertes Marrón, Moungi G Bawendi et al. 2020. "How machine learning can help select capping layers to suppress perovskite degradation." *Nature communications* 11 (1), p. 4172

, "Template Matching." https://docs.opencv.org/3.4/d4/dc6/tutorial_py_template_matching.html

Appendices