Data Science Intern at Data Glacier

Corrosion Detection & Severity Level Prediction Using Machine Learning

Week 12

Final Report.

Name: Syed Sanaullah Shah

University: NUST, Islamabad

Email: iamkadhimi@gmail.com

Country: Pakistan

Specialization: Data Science

Batch Code: LISUM11

Date: 30 October 2022

Submitted to: Data Glacier

Table of Contents:

Problem Statement

Corrosion of pipelines has been recognized as a major predicament because it undermines the mechanical strength of pipelines, which can pose a disaster for the environment and the economy. Most of the time, the O&G industry uses mild steels. Ultrasonic testing during accelerated corrosion testing is a way to detect corrosion. AI is already using the scientific approach by popping up with hypotheses and doing simple tests to see if they are true. In the mid to long term, AI could have a big impact on how research is conducted, constructed, and acknowledged. However, using machine learning techniques to classify corrosion based on parameters like weight and thickness loss is still a new idea.

Our framework suggests using a new method that looks for corrosion through thickness and mass losses by NDT. With the help of machine learning methods and ultrasonic testing from accelerated corrosion testing, corrosion severity levels can be predicted based on data generated in the lab. Samples of mild steel could be evaluated for accelerated corrosion over four different time periods in a lab. Samples' thickness and mass loss were noted down in the dataset. Ultrasonic testing readings were done with an Epoch LT Ultrasonic testing 4

machine with a straight beam configuration, because the relationship between thickness and mass losses and the corrosion process is linear. For multi-class problem, four Corrosion severity levels have been created based on thickness & mass loss occurred during accelerated corrosion testing for which XGB, SVM and Random Forests showed cross validation accuracy score of 99.8%, 93.5%, and 91.5% respectively.

Data Understanding

Data acquisition was the first objective of our project where we simulated pipeline corrosive conditions to develop a framework where we could access the risk analysis by measuring the features that contribute to corrosion in oil and gas pipelines. Now that the first step has been achieved, it was time to go to the next phase of our project where programming and machine learning expertise were utilized extensively. The machine learning phase included data preparation, selection of machine learning methods, feature engineering and model deployment after final hyperparameter optimizations.

The results of the data generated via lab experiments are recorded. Dataset contains 11 columns and 48 data points from 12 samples. Every mild steel sample gave us 4 datapoints. The explanation of each column is as follows:

- Data Point: The unique unit of information extracted from lab experiments.
- Target: The labeling of output.
- Before[g]: The weight of sample recorded before corrosion.
- After[g]: The weight of sample recorded after corrosion.
- Weight Loss[g]: The sample wight loss in the recorded time period. It is obtained by subtracting after[g] from before[g].
- %Weight Loss: The percentage of weight loss. It is the obtained by dividing Weight Loss[g] and Before[g] then multiplying with 100.
- Before[mm]: The weight of sample recorded before corrosion
- After[mm]: The weight of sample recorded after corrosion.
- %Thickness Loss: The percentage of weight loss. It is the obtained by dividing Weight Loss[g] and Before[g] then multiplying with 100.
- Time: It denoted the corrosion time of the sample in the laboratory.

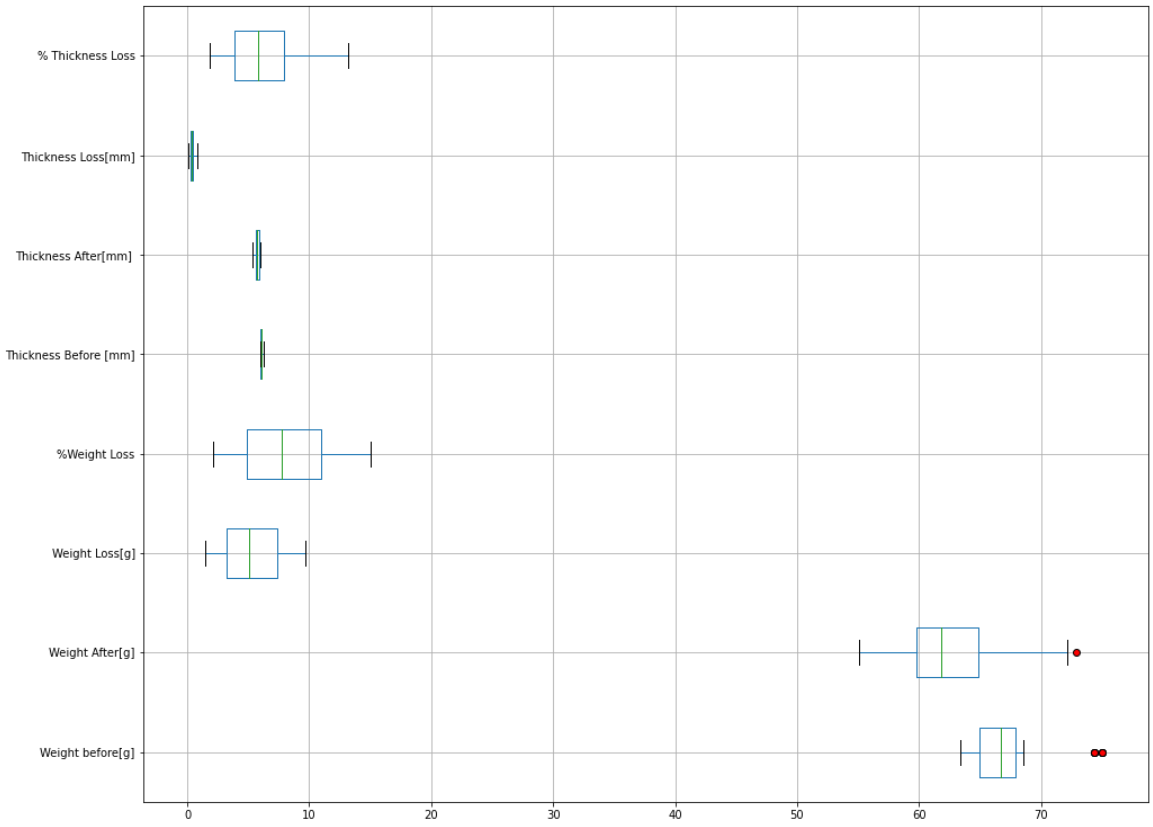
Data Preprocessing

The data cleaning process detects and removes the errors and inconsistencies present in the data and improves its quality. We solve this by eliminating missing values, noise reduction and outliers removal. Replace missing values by the mean or median or simply putting 0 in missing datapoints.

Outliers can be reduced by the use of binning methodology to smoothen the affects of outliers. Pandas is effectively used in cleaning the dataset and in the process of reckoning the outliers.

The major issue in our dataset is the lack of data as more data would be helpful. Since this data is generated in the lab with only handful of experiments thus the possibilities of data problems are negligible to non existant.

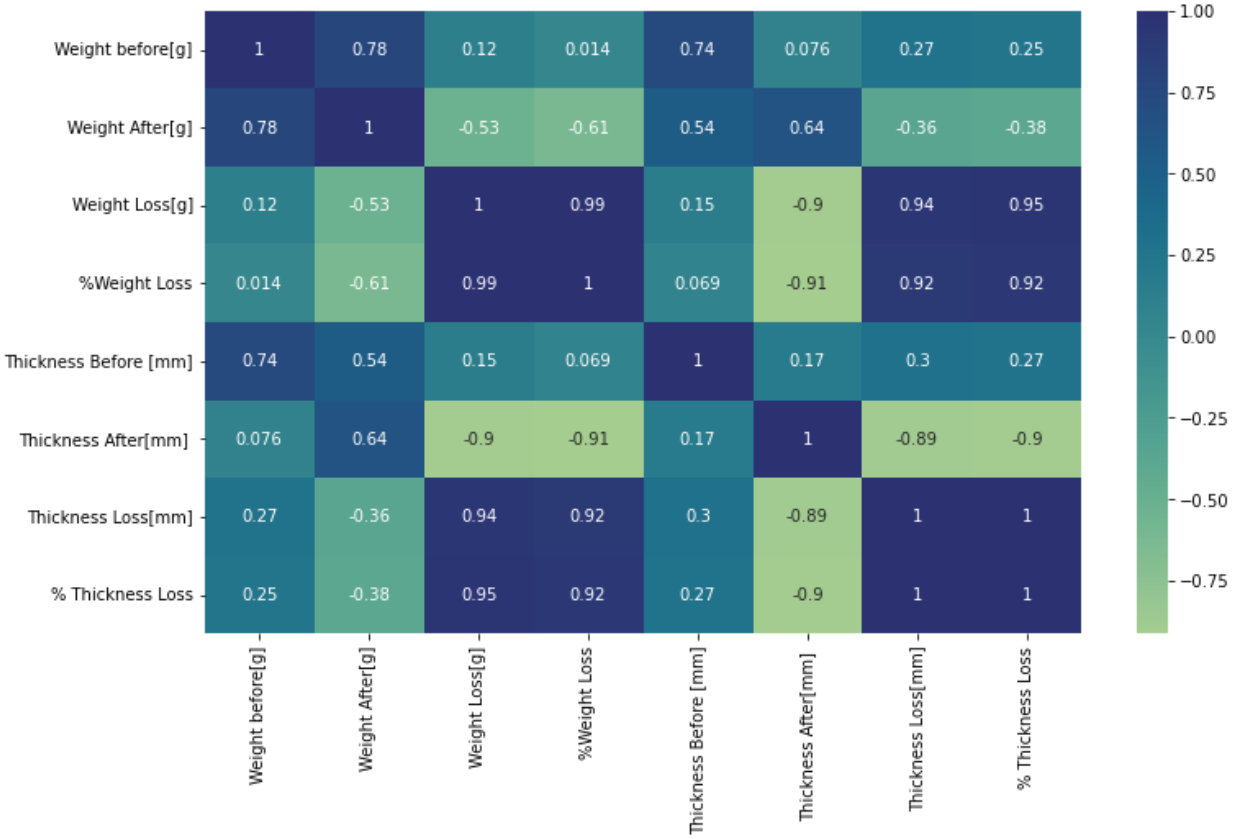
The dataset contains 48 instances, which is little by Machine Learning standards, but it's sufficient for getting started. All attributes are numerical with no repeating instances that save us the time consumed by categorical features encoding. The first step was to remove any duplicates and look for null-values. However, these were not observed in any of the features. Moving forward, we stat the results of exploratory data analysis. The initial investigation to understand the data and detect anomalies is to check for outlier detection using box plot graphical representation.



Outliers Detection

In our data set except “Weight before(g) and Weight After(g)”, no other features columns show outliers. This can be explained due to excessive weight loss that occurred due to increase acidity of the corrosion cell. Since the solution was reused more than once, that is why, a couple of samples experienced extravagant loss in wight.

The correlation between two random variables is a value that ranges from -1 to +1, indicating a strong inverse association, no relationship, or a strong direct relationship. We found correlations using pandas “.corr()” function and visualized the correlation matrix using a heatmap in seaborn. We used Pearson correlation that evaluates the linear relationship between two continuous variables. Here dark shades represent positive correlation while lighter shades represent negative correlation. Interesting insights were obtained. We can see that wight before(g) is strongly associated with weight after(g) and thickness before(mm). We can also deduce that weight loss has direct correlation with %weight loss, thickness loss(mm), and %thickness loss. Furthermore, as suspected, there seems to be direct relationship between thickness loss(mm) and weight & %weight loss as well.

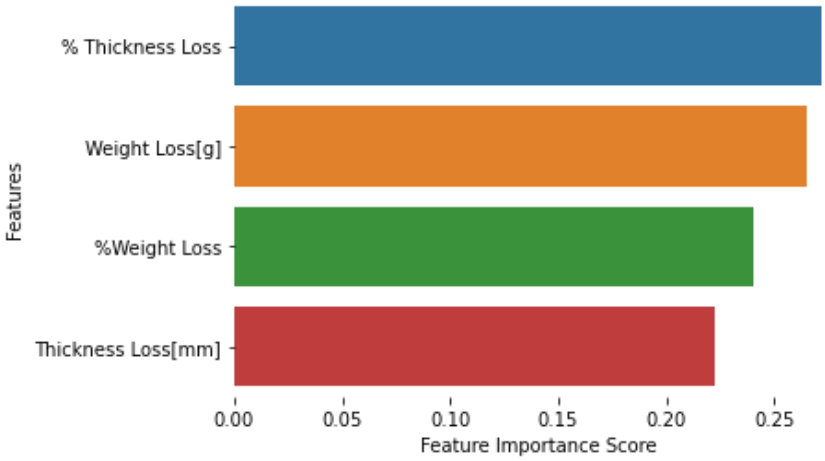


Correlation Matrix

Therefore, we now have some understanding of our data. The next step is to find and extract relevant features from our dataset to use in classification in the feature engineering step.

Feature Engineering

After looking at relationships between various features, it's time to select only relevant features that had the most impact on the model performance. More precisely how accurate they will classify the classes in testing phase Tree derived feature importance is a very straightforward, fast and accurate method of selecting suitable features for machine learning. Thus. embedded methods combine the qualities of filter and wrapper methods is used in feature engineering step. We utilized random forests algorithm to extract key features. According to the feature relevance score of the input variables, the percentage thickness loss has the greatest influence on the target feature, followed by weight loss[g], thickness loss[mm], and percent weight loss. High score indicates that it plays a big part in predicting the outcome. In short, we cannot include every feature in modelling as that would dull the performance of the model.



Feature Importance

CLASSIFICATION RESULTS

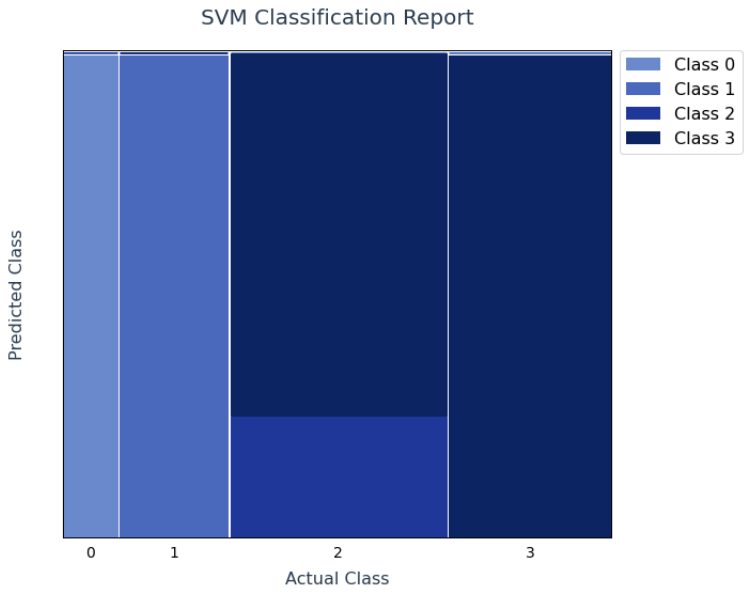
The performance of the proposed machine learning approaches using the RF and the LS-SVM classifiers in predicting discrete levels of gasification products is evaluated by computing commonly used evaluation measures for binary classification referred to as classification accuracy (A), precision (P), recall (R) and F1-score, squared r and root mean squared error

Classification report results are illustrated via Mosaic diagram that is a different type of stacked bar chart. One can easily see the predicted classes on the y-axis and the number proportion of each class on the x-axis. What's special here from a simple bar plot is the width of the bars, which are giving an idea of the observed class imbalance. Unlike heatmap, it is easy to show relationships and to provide a visual comparison of discrete classes. The performance of classifiers includes the best parameters selected, precision, recall, f1-score, R^2 error, and RMSE values. In the end, every classifier's mosaic diagram further explains the performance graphically.

Support Vector Machine

C is the penalty parameter and helps in fitting the boundaries smoothly with default=1. Kernel is a similarity function for pattern analysis. It must be one of the rbf /linear /poly /sigmoid. Choosing an appropriate kernel will result in a better model fit. While gamma is the curvature weight of the decision boundary. Optimized parameters include C=1, gamma='scale' and kernel='rbf'. SVM gave the accuracy score of 0.70. Classification report is shown in table. Mosaic diagram shows that Grade C class was inaccurately classified as Grade D that amounts to inaccuracy in support vector algorithm.

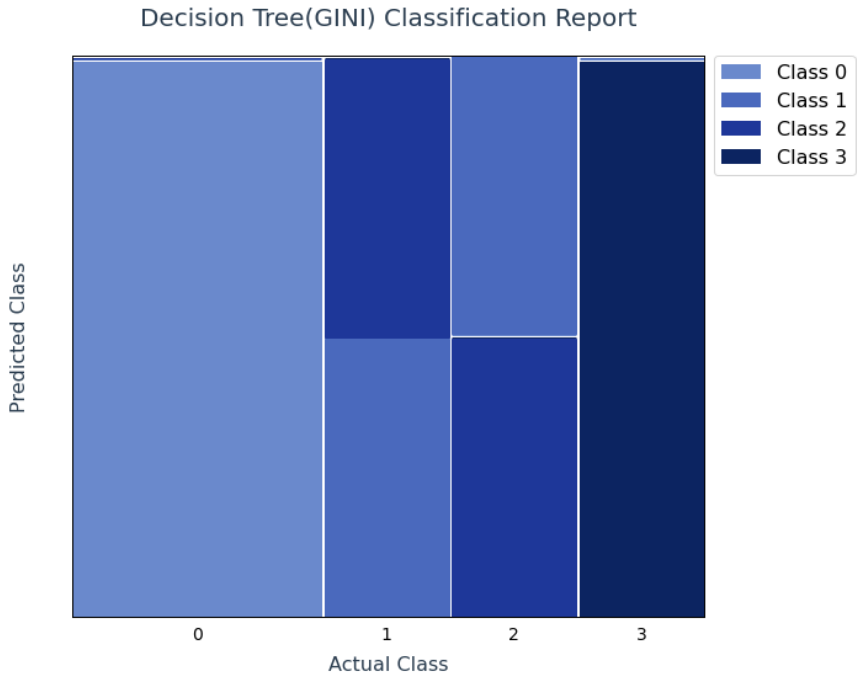
	precision	recall	f1-score	support
Grade A	1.00	1.00	1.00	1
Grade B	1.00	1.00	1.00	2
Grade C	1.00	0.25	0.40	4
Grade D	0.50	1.00	0.67	3
accuracy			0.70	10
macro avg	0.88	0.81	0.77	10
weighted avg	0.85	0.70	0.66	10



Decision Tree (Gini)

Decision tree various parameters include max_features that is the maximum features to be considered while deciding each split, min_samples_split that is the split will not be allowed for nodes that do not meet this number, min_samples_leaf tell us that leaf node will not be allowed for nodes less than the minimum samples and max_depth inform us that no further split will be allowed. With criterion information gain, only max_depth=3 was considered while others were kept as default. DT gave us accuracy score of 0.80. Classification report is in the table. Mosaic diagram enlightens us here about the inaccuracies in classification where few grade B were classified as Class C and vice versa.

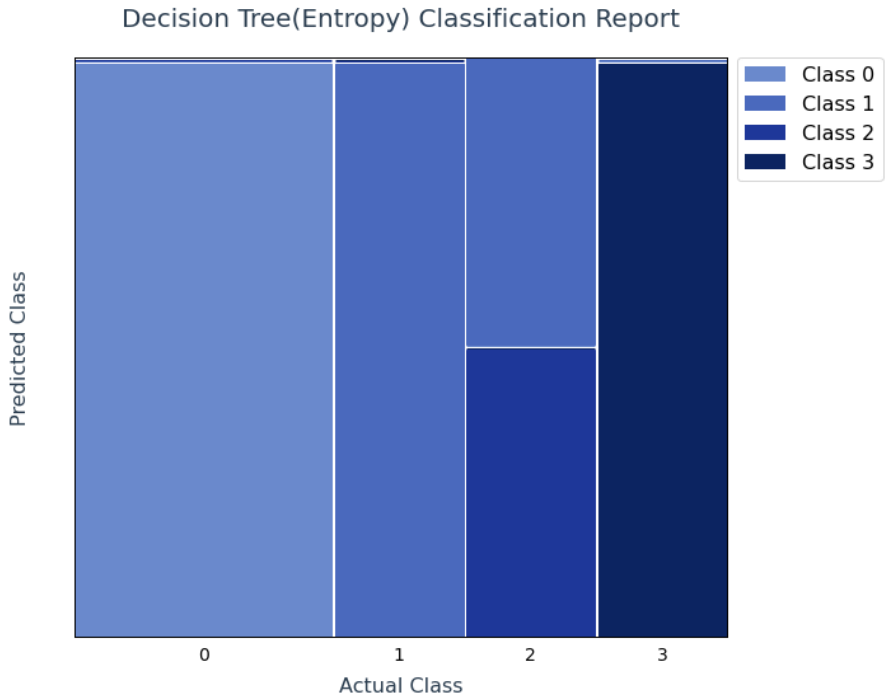
	precision	recall	f1-score	support
Grade A	1.00	1.00	1.00	4
Grade B	0.50	0.50	0.50	2
Grade C	0.50	0.50	0.50	2
Grade D	1.00	1.00	1.00	2
accuracy			0.80	10
macro avg	0.75	0.75	0.75	10
weighted avg	0.80	0.80	0.80	10



Decision Tree (Entropy)

Decision tree various parameters include max_features that is the maximum features to be considered while deciding each split, min_samples_split that is the split will not be allowed for nodes that do not meet this number, min_samples_leaf tell us that leaf node will not be allowed for nodes less than the minimum samples and max_depth inform us that no further split will be allowed. With criterion entropy, only max_depth=3 was considered while others were kept as default. DT gave us accuracy score of 0.90. Classification report is in the table. Mosaic diagram enlightens us here about the inaccuracies in classification where few grade C were classified as Class B.

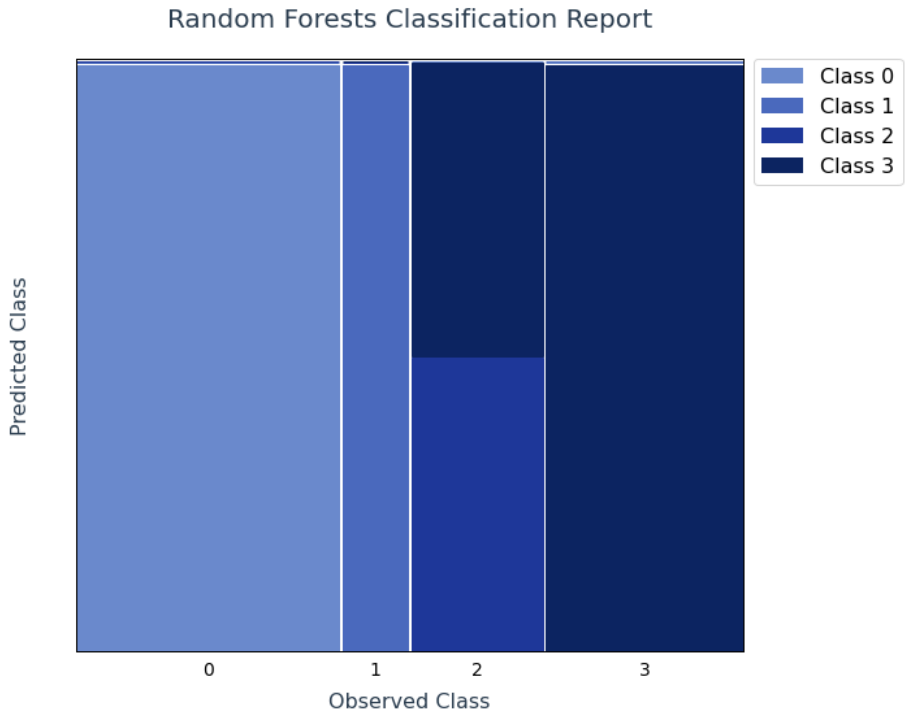
	precision	recall	f1-score	support
Grade A	1.00	1.00	1.00	4
Grade B	0.67	1.00	0.80	2
Grade C	1.00	0.50	0.67	2
Grade D	1.00	1.00	1.00	2
accuracy			0.90	10
macro avg	0.92	0.88	0.87	10
weighted avg	0.93	0.90	0.89	10



Random Forests

Rndom forests parameters include n_estimators that is the number of weak learners to be built, max_depth is the maximum depth of the individual estimators. Here, we only considered 100 weak leaners. RF gave the accuracy score of 0.90. Classification report is as follows: Mosaic report shows few instances of grade C were classified as Grade D.

	precision	recall	f1-score	support
1	1.00	1.00	1.00	4
2	1.00	1.00	1.00	1
3	1.00	0.50	0.67	2
4	0.75	1.00	0.86	3
accuracy			0.90	10
macro avg	0.94	0.88	0.88	10
weighted avg	0.93	0.90	0.89	10



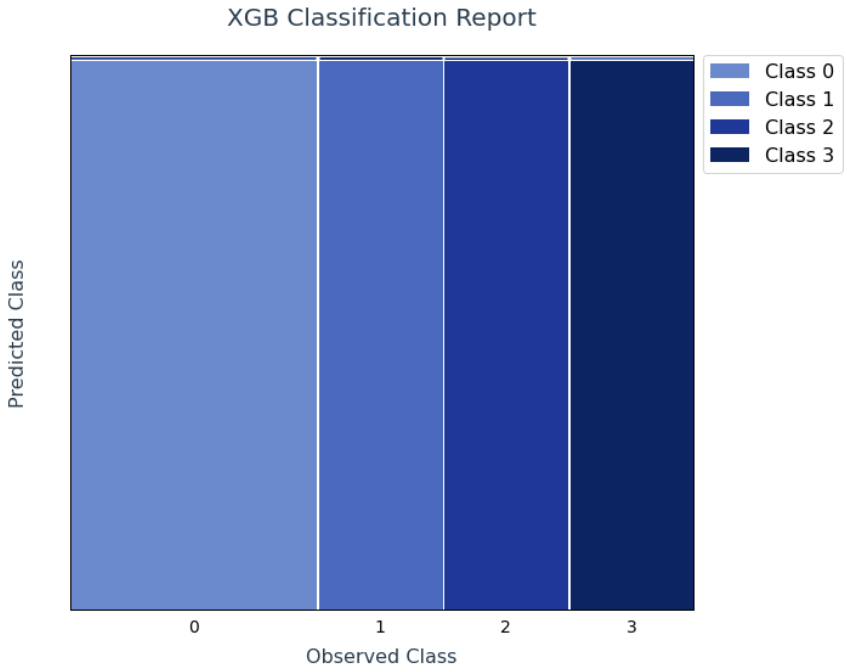
XGBOOST

XGboost has a bunch of parameters & we can group them into three categories that are General Parameters, Boosting Parameters, and Task Parameters. We are only concerned with boosting parameters. Some of them are as follows:

- Learning rate(eta): This is the learning rate or step size shrinkage to prevent over-fitting; default is 0.3 and it can range between 0 to 1.
- max_depth: Maximum depth of tree with default being 6.
- Gamma: Minimum loss reduction required to make a further partition on a leaf node of the tree.
- min_child_weight: Minimum sum of weights of all observations required in child. Start with 1/square root of event rate
- colsample_bytree: Fraction of columns to be randomly sampled for each tree with default value of 1.
- Subsample: Fraction of observations to be randomly sampled for each tree with default of value of 1. Lowering this value makes algorithm conservative to avoid overfitting.
- Lambda: L2 regularization term on weights with default value of 1.
- Alpha: L1 regularization term on weight.

We tuned learning rate= 0.25, n-estimators=400, max_depth=4 and lastly gamma=0.3 while keeping other parameters as default. Learning rate and trees proportion is inversely related for optimal accuracy. XGB gave the accuracy score of 1. Classification report is as follows: Mosaic diagram has shown that every class was classified accurately.

	precision	recall	f1-score	support
Grade A	1.00	1.00	1.00	4
Grade B	1.00	1.00	1.00	2
Grade C	1.00	1.00	1.00	2
Grade D	1.00	1.00	1.00	2
accuracy			1.00	10
macro avg	1.00	1.00	1.00	10
weighted avg	1.00	1.00	1.00	10



4.8 Graphical User Interface

Graphical User Interface allows users to interact with electronic equipment utilizing graphical icons, symbols, and user-friendly software with a command-driven interface. The

GUI presented in this paper allowed users to input data of thickness and mass losses along their mass loss percentages. GUI uses the Xgboost model prediction function to predict corrosion severity levels. GUI was developed in Streamlit that is an open-source Python library. Figure illustrates GUI where a random sample data is used to predict corrosion severity level. Mass loss and mass percentage loss was recorded as (6.91 & 10.3%) whereas thickness loss and thickness percentage loss (0.38 & 6.2) were inserted as an input and model was operated with the help of push button. GUI predicted the grade C level of corrosion.

XGB GUI of Web App

Summary of Comparative study

The following is a comparative analysis of the supervised classification algorithms that are both commonly utilized and highly sought after. The confusion matrix, R-squared, and root mean squared error are the metrics that are used to evaluate accuracy. Rather than focusing solely on a deep analysis, this chapter gave an overview of the many supervised learning methods along with the optimized parameters that determine the level of accuracy achieved.

We found that Xgboost was the method that performed the best when classifying tabular data, followed by a different ensemble bagging methodology and support vector machines.

Table: Hyperparameter Optimization Summary

ML Methods	Parameters	Ranges	Optimized Values	Cross Val Score(10 F)
Decision Tress	Criterion Max-Depth	GINI 3-5	GINI 3	0.855
Decision Tress	Criterion Max-Depth	Entropy 3-5	Entropy 3	0.895
SVM	'C' 'Gamma' 'Kernel' 'CV'	0.1-100 'scale'- 0.001 'RBF', 'POLY' 3-10	1 'scale' 'RBF' 10	0.935
Random Forests	Decision Trees	10-100	10	0.915
XGB	Max-Depth Learning Rate 'n-estimators' 'Gamma;	3-15 0.01-.3 10-500 0-0.4	4 0.25 400 0.3	0.998

The summary of the results of the evaluation metrics reveals the better classification results for ensemble techniques. Xgboost and random forests have shown the least deviations in predicted values from actual values. The reason the Xgboost is better at predicting than the RF is that it gives higher P and R values for each product, which leads to higher F1-scores. Also, the Xgboost can get F1-scores of 1, while the RF could only get F1-scores of 0.88. We employed R-squared and RMSE, which resonates with the results of F1-score, in order to

further examine the efficacy of the machine learning classifiers in determining different levels of corrosion. This allowed us to more accurately assess the performance of the classifiers.

Table: Evaluation Metrics Summary

ML Methods	Precision	Recall	F1	Accuracy	R Squared	RMSE
Decision Tress(GINI)	0.75	0.75	0.75	0.8	0.852	0.447
Decision Tress(Entropy)	0.92	0.88	0.87	0.9	0.926	0.316
SVM	0.88	0.81	0.77	0.70	0.663	0.548
Random Forests	0,94	0.88	0.88	0.9	0.852	0.447
XGB	1	1	1	1	1	0

Conclusion

Xgboost fared better than other classifiers in predicting corrosion severity level. To summarize, we have achieved our objectives of data collection & comparative study of best performing machine learning method for corrosion severity detection.

CODE

Final Project:

1. https://github.com/kaz912r/Final_Project_DG/blob/main/Week9-11.ipynb
2. https://github.com/kaz912r/Final_Project_DG/blob/main/Week12.ipynb

PRESENTATION

1. https://github.com/kaz912r/Final_Project_DG/blob/main/Week12_Final%20project_Deliverables.pdf