Prediction of Erosion Rate in Pipelines Using Machine Learning

Author: Shouq Alrumaih

July 2025

Introduction

In computational fluid research (CFR) and industrial pipeline monitoring, manually calculating erosion rates can be time consuming and complex. Machine learning offers fast and relatively less effort alternative that can be trained once and used repeatedly for accurate predictions across varying conditions.

In oil and gas pipelines, especially at bends and elbows, erosion from solid particles like sand can lead to severe damage. Predicting this erosion helps minimize failures, maintenance costs, and unplanned shutdowns. This report outlines how we applied machine learning (ML) techniques to predict erosion rates using structured experimental data.

This work builds upon a previous study [1] that applied a Random Forest model to predict erosion under gas-sand and multiphase flow conditions. Using the same dataset and a reproducible evaluation setup, we explored additional models such as Gradient Boosting, and applied preprocessing, tuning, and data splitting strategies.

Our Gradient Boosting and Random Forest models achieved higher accuracy and lower error metrics than the original Random Forest model reported in the article, suggesting that further optimization can significantly improve predictive performance in erosion modeling.

Data Preparation

We worked with two separate datasets provided in the article [1]:

Gas-sand flow experiments (Table 4) and Multiphase flow experiments (Table 5)

We combined them into a single dataset, then performed the following preprocessing steps:

Removed unused columns (Reference, Predicted Erosion, Error%)

Filled missing values in Liquid Viscosity and Liquid Superficial Velocity with zeros

One-hot encoded the Pipe Wall Material column, since models cannot interpret text. We used pandas.get_dummies[3], assigning a separate column to each category (1 = present, 0 = not), to avoid misleading the model into interpreting label values as ordinal.

Added a splitter column to indicate the source (0 = gas-sand, 1 = multiphase)

Used stratified train-test splitting (50/50) [2] based on splitter column to ensure fair representation from both datasets

Feature Scaling

Some models are sensitive to feature scale. For instance, Particle Size ranges up to 300, while Pipe Diameter is around 0.07. To prevent larger features from dominating learning, we used StandardScaler [2] to normalize all features.
Scaling was especially important for:

Support Vector Regressor (SVR): Relies on distance-based kernels.

Neural Network (MLP): Gradient-based learning is more stable with standardized input.

Model Selection

Models were built and evaluated using scikit-learn [2]. We evaluated the following models:

Linear Regression: This model can be seen as fitting a straight line through the data. It assumes a linear relationship between the input features and the target variable and aims to minimize the squared error between predicted and actual values.

Random Forest Regressor (RF): This model aggregates predictions from many decision trees, each trained on different subsets of the data. By averaging their outputs, it reduces overfitting and improves overall prediction accuracy.

Support Vector Regressor (SVR): SVR attempts to fit a function within a specified margin of tolerance. It uses only the critical data points that lie outside this margin called support vectors and applies kernel methods to capture non-linear relationships.

Neural Network (MLPRegressor): A neural network consists of multiple layers of connected neurons that learn complex non-linear patterns in the data. Through iterative weight adjustments during training, it captures intricate dependencies between inputs and outputs.

Gradient Boosting Regressor (GB): This model builds decision trees sequentially, where each tree is trained to correct the residual errors of the previous ones. It gradually improves accuracy by focusing on the parts the earlier models got wrong.

Evaluation metrics:

Mean Absolute Error (MAE): Used to measure the average size of the errors in a set of predictions. We used it to understand how far off our predictions were on average.[4]

$$\mathrm{MAE} = \frac{1}{\mathrm{N}} \sum_{\mathrm{j}=1}^{\mathrm{N}} |\mathrm{y_j} - \hat{\mathrm{y}}_\mathrm{j}|$$

- y_j = Actual value
- $\hat{y}_j$ = Predicted value

Mean Squared Error (MSE): Measures the average of the squares of the errors. Squaring means larger errors are penalized more heavily, which helps us detect models that occasionally make large mistakes.[4]

$$\mathrm{MSE} = \frac{1}{\mathrm{N}} \sum_{\mathrm{j}=1}^{\mathrm{N}} (\mathrm{y_j} - \hat{\mathrm{y}}_\mathrm{j})^2$$

- y_j = Actual value
- $\hat{y}_j$ = Predicted value

R^2 Score (coefficient of determination): Evaluates how much of the target variable's variance is explained by the model. A score closer to 1.0 means the model fits the data well. This helped us determine which model best captured the patterns in erosion behavior.[4]

$$R^2 = 1 - \frac{\sum_{j=1}^{n} (y_j - \hat{y}_j)^2}{\sum_{j=1}^{n} (y_j - \bar{y})^2}$$

- y_j = Actual value
- $\hat{y}_j$ = Predicted value
- $\bar{y}$= Mean of the actual values

Models Evaluation Results

We applied hyperparameter tuning to improve model performance. Hyperparameters are settings chosen before training that affect how the model learns like learning rate, number of neurons, gamma, or kernel type. They can influence how fast the model trains and how well it performs.

To tune them, we used GridSearchCV [2], which tries all combinations of selected hyperparameter values and uses cross-validation to find the best setup. Although it can be slow, it often leads to better results.

In our case:

SVR and Neural Networks improved significantly after tuning.

Random Forest and Gradient Boosting performed better with default settings.

Model	MAE (Default)	MAE (Tuned)	R^2 (Default)	R^2 (Tuned)	MSE (Default)	MSE (Tuned)	Observation
Linear Regression	0.0139		0.7923		0.00032		
SVR	0.0960	0.0072	-5.01	0.9359	0.00933	9.95e-05	Huge improvement after tuning
Neural Network	0.054	0.0117	-10.1	0.7851	0.01732	0.0003	Huge improvement after tuning
Random Forest	0.001939	0.0023	0.9801	0.9711	3.086235 e-05	4.471e-05	Performed better with default settings
Gradient Boosting	0.001747	0.0021	0.9830	0.9700	2.6274e-05	4.652e-05	Performed better with default settings

Table 1. Models Results

As we have a small dataset (202 rows), the models were trained on half of the data and tested on the other half. Random Forest (see Figure 1) performed very well, likely due to its robustness and ability to handle small datasets by averaging across multiple trees, which reduces overfitting and helps it generalize even with limited data. While Neural Network are typically powerful, underperformed because it usually requires large datasets to train effectively and avoid instability. Linear Regression didn't perform well because it assumes a purely linear relationship between inputs and outputs, while our data might exhibits more complex, non-linear patterns.

SVR also didn't perform strongly likely due to the noise in data or because we didn't do much feature engineering which is crucial for SVR. Overall, Random Forest stood out as the most stable performer under our limited-data constraints, thanks to its ensemble nature that merges multiple decision trees, leading to more accurate predictions than single models. This ensemble approach is a key factor in its success.

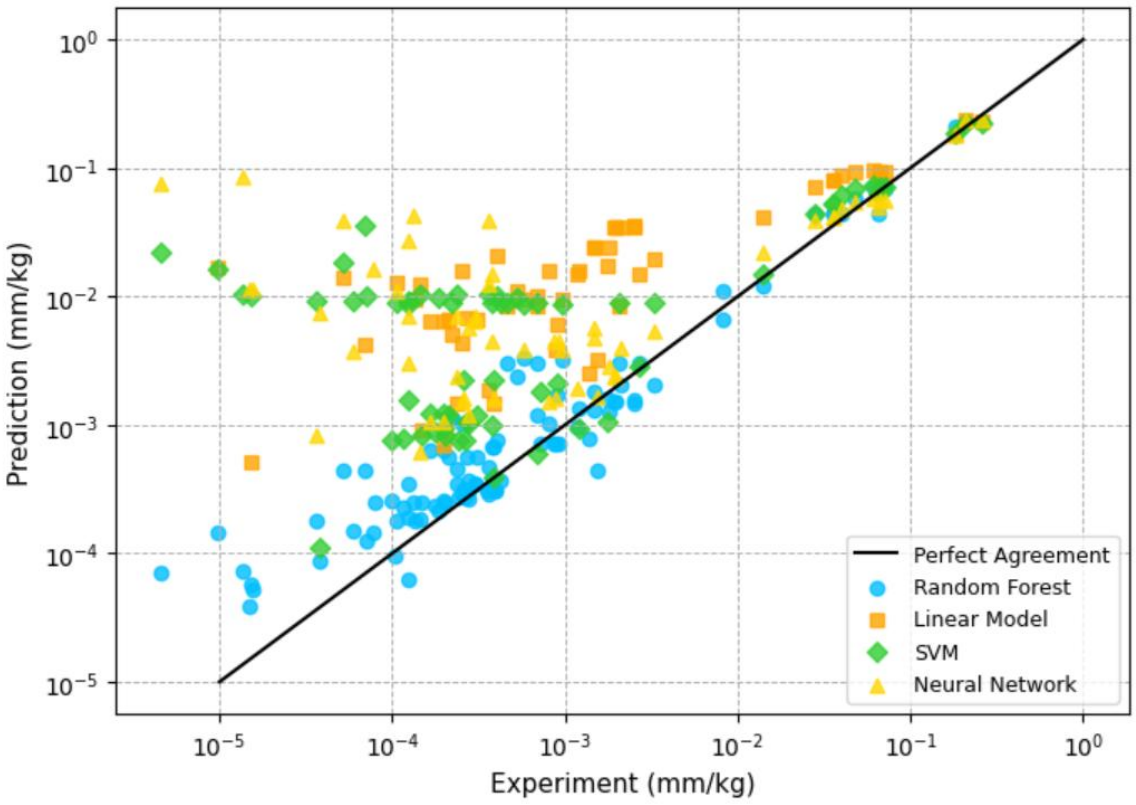

Figure 1. Predictions – RF, LM, SVR, NN models With Stratify Splitting

Although Random Forest is often praised for its robustness in noisy or limited datasets, in our case Gradient Boosting slightly outperformed it (see Table 1). This result can be attributed to Gradient Boosting's sequential learning approach, which focuses on correcting previous errors and can capture more subtle, complex patterns.

Given that our dataset, while small, still exhibited some structure, Gradient Boosting was able to adaptively fit the data more precisely than Random Forest leading to better predictive performance (see Figure 2).

That said, Random Forest still showed strong and stable performance, which aligns with its reputation for robustness and ease of use.

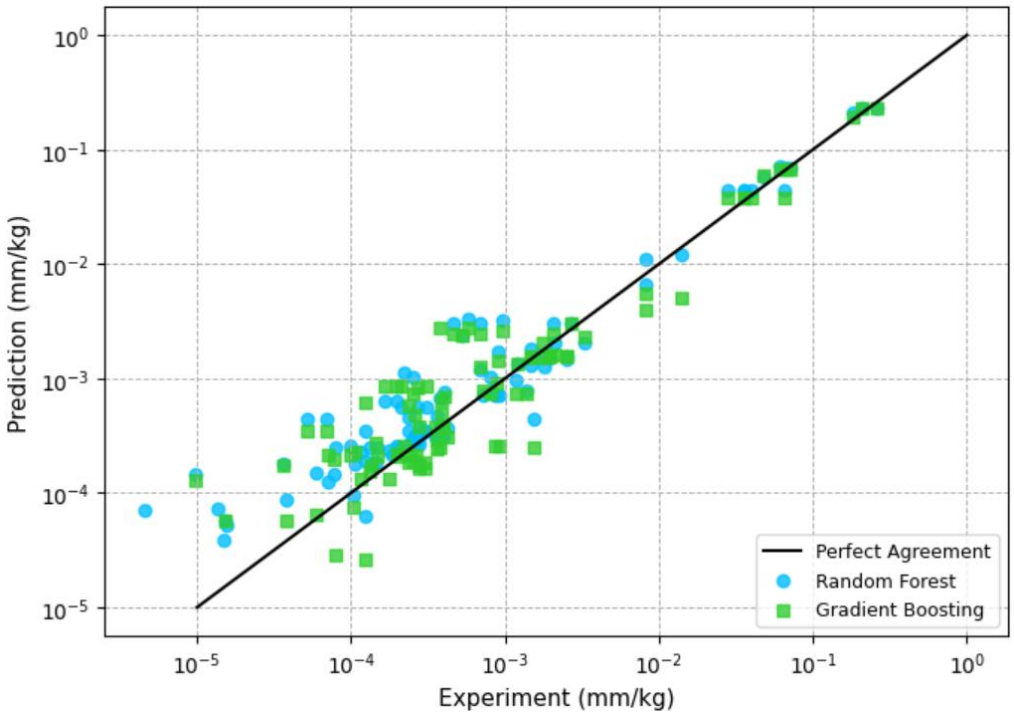

Figure 2. Predictions – RF vs GB With Stratify Splitting

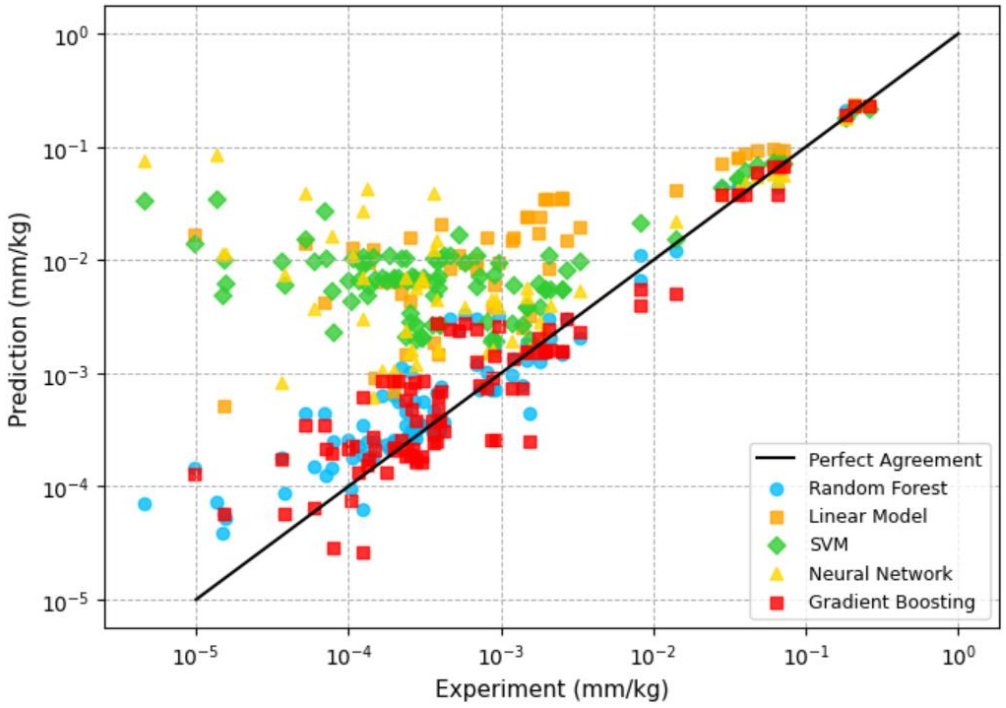

Figure 3. Final Predictions for all models with Stratify Splitting

Label Inconsistency

During the data validation phase a significant quality issue was identified. 31 unique input combinations appeared multiple times in the dataset with conflicting Measured Erosion values affecting a total of 74 rows. These combinations consisted of identical values across all input features, including pipe wall material, pipe diameter, particle size, liquid viscosity, liquid superficial velocity, and gas superficial velocity.

This issue violates a fundamental assumption of supervised learning: that the same input features should consistently correspond to the same target output. When identical feature vectors are associated with varying target values, it introduces ambiguity into the learning process, making it difficult for the model to converge on a reliable mapping function.

Example:

For instance, the following configuration appears repeatedly with different erosion outcomes:

```
Pipe Wall Material: ASTM 234
Pipe Diameter: 0.0508 m
Particle Size: 550 µm
Liquid Viscosity: 0 cP
Liquid Superficial Velocity: 0.00 m/s
Gas Superficial Velocity: 108.0 m/s
Measured Erosion values: ranging from 0.0354 to 0.0807 mm/kg
```

Such inconsistencies are likely due to either missing hidden variables, experimental noise, or measurement inaccuracies. They directly affect the model's ability to generalize and may partially explain the degraded performance observed during model evaluation.

Figure 4 illustrates a sample of the affected rows where identical input features are linked to divergent measured erosion values.

Number of confusing input patterns with conflicting target values: 31
Total rows affected: 74

	Pipe Wall Material	Pipe Diameter (m)	Particle Size (μm)	Liquid Viscosity (cP)	Liquid Superficial Velocity (m/s)	Gas Superficial Velocity (m/s)	Measured Erosion (mm/kg)
51	ASTM 234	0.0508	550.0	0.0	0.00	98.0	0.035400
52	ASTM 234	0.0508	550.0	0.0	0.00	98.0	0.035500
49	ASTM 234	0.0508	550.0	0.0	0.00	103.0	0.037600
50	ASTM 234	0.0508	550.0	0.0	0.00	103.0	0.039700
47	ASTM 234	0.0508	550.0	0.0	0.00	107.0	0.048200
48	ASTM 234	0.0508	550.0	0.0	0.00	107.0	0.059300
43	ASTM 234	0.0508	550.0	0.0	0.00	108.0	0.071200
44	ASTM 234	0.0508	550.0	0.0	0.00	108.0	0.055100
45	ASTM 234	0.0508	550.0	0.0	0.00	108.0	0.080700
46	ASTM 234	0.0508	550.0	0.0	0.00	108.0	0.066000
41	ASTM 234	0.0508	550.0	0.0	0.00	177.0	0.238000
42	ASTM 234	0.0508	550.0	0.0	0.00	177.0	0.253000
35	CS-100_95	0.0254	150.0	1.0	0.03	27.4	0.000694
37	CS-100_95	0.0254	150.0	1.0	0.03	27.4	0.002060
40	CS-100_95	0.0254	150.0	1.0	0.03	27.4	0.000460
36	CS-100_95	0.0254	150.0	1.0	0.30	27.4	0.000873
38	CS-100_95	0.0254	150.0	1.0	0.30	27.4	0.006880
39	CS-100_95	0.0254	150.0	1.0	0.30	27.4	0.000576
53	SS-316	0.0762	300.0	0.0	0.00	15.0	0.000355
54	SS-316	0.0762	300.0	0.0	0.00	15.0	0.000393
55	SS-316	0.0762	300.0	0.0	0.00	15.0	0.000276

Figure 4. Inconsistent Target Labels Across Duplicated Input Records

To address this issue, we applied an aggregation strategy by computing the average Measured Erosion value for each group of duplicated input records. Once the values were averaged, the duplicate rows were removed to retain only one representative instance per unique input configuration. This step ensures that the model is trained on a more deterministic dataset, where each unique combination of features corresponds to a single, consistent target value.

As a result of this preprocessing, the dataset was reduced from 202 rows to 159 rows. This modification could improve the reliability of the supervised learning setup and is expected to enhance model stability and predictive accuracy in evaluations. But the results were not satisfactory, as shown in Figure 5.

Instead of averaging, we applied a selective filtering approach: for each group of duplicate records with high variation in Measured Erosion, we retained only the value closest to the group's median. This allowed us to reduce noise without losing much data. After this step, the dataset contained 197 rows, but model performance remained unchanged. This confirms that the limited dataset size and inconsistencies in target labels are the main factors hindering further improvement.

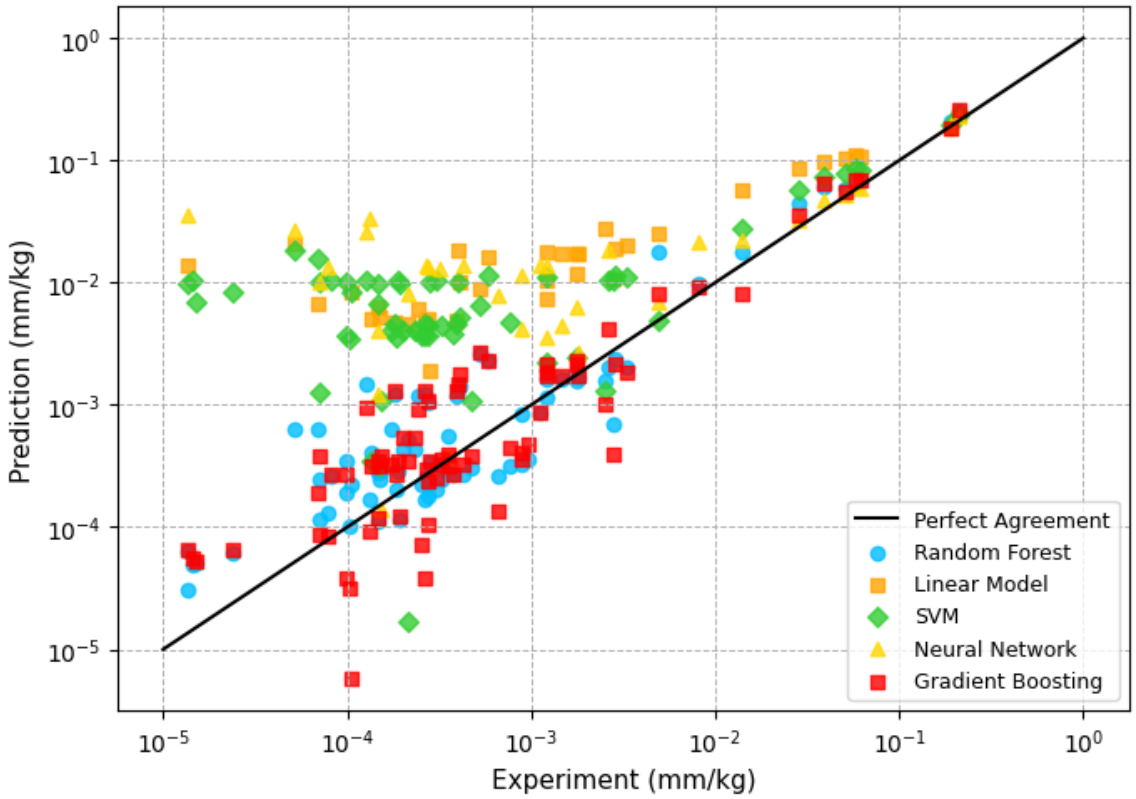

Figure 5. After Solving Inconsistent Target Labels

Back to data preparation

While the article [1] did not specify the method used to split the dataset we assume it was performed using a random split, which is the common default in machine learning studies. In contrast, we initially applied stratified splitting to ensure balanced representation from both flow regimes (gas-sand and multiphase) in the train and test sets.

To evaluate the effect of the splitting strategy, we later tested our models using a random 50/50 split similar to the presumed approach in the article Figure 8. The results showed that Random Forest, Gradient Boosting and SVR achieved improved performance in terms of Mean Squared Error (MSE) and Mean Absolute Error (MAE) under the random split, as shown in Table 2.

This indicates that splitting strategy can have a notable impact on evaluation metrics and in our case the random split slightly benefited these 3 models.

Model	MAE	R^2	MSE
Article's Random Forest	0.00150	0.989	2.357e-05
Our Random Forest	0.00145	0.985	1.576e-05
Our Gradient Boosting	0.00149	0.980	2.1656e-05
Our Support Vector Regressor	0.0065	0.939	6.834e-05

Table 2. Results After Random Split On Half Of Data

We cannot confirm whether "Predicted Erosion" column [1] were calculated using models trained on the full dataset or on a subset of the data. However, if we assume that their "Predicted Erosion" column reflects a model trained and evaluated on the entire dataset, then it is appropriate to assess our models under the same conditions for a fair comparison.

Accordingly, we retrained both the Random Forest and Gradient Boosting models on our full dataset and evaluated them directly on that same data. As shown in Table 3 and Figure 6, both models achieved improved performance.

Model	MAE	R^2	MSE
Article's Random Forest	0.00150	0.989	2.357e-05
Our Random Forest	0.00118	0.988	2.698e-05
Our Gradient Boosting	0.0006	**0.998**	2.979**e-06**

Table 3. Results On The Full Dataset

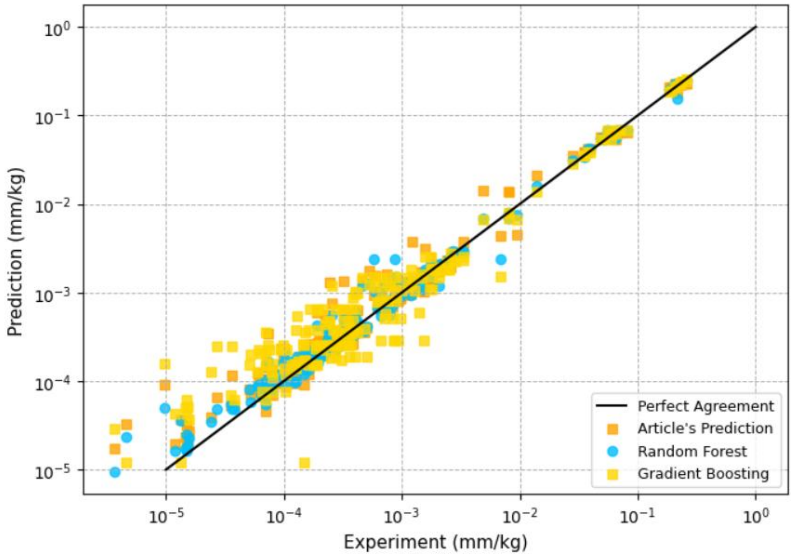

Figure 6. Results On The Full Dataset

Returning to our splitting strategy, if the goal is to present each model under conditions where it performs best, we found that stratified splitting generally yielded better results for Linear Regression and Neural Networks, while random splitting led to improved performance for Support Vector Regression, Random Forest, and Gradient Boosting, as illustrated in Figure 7.

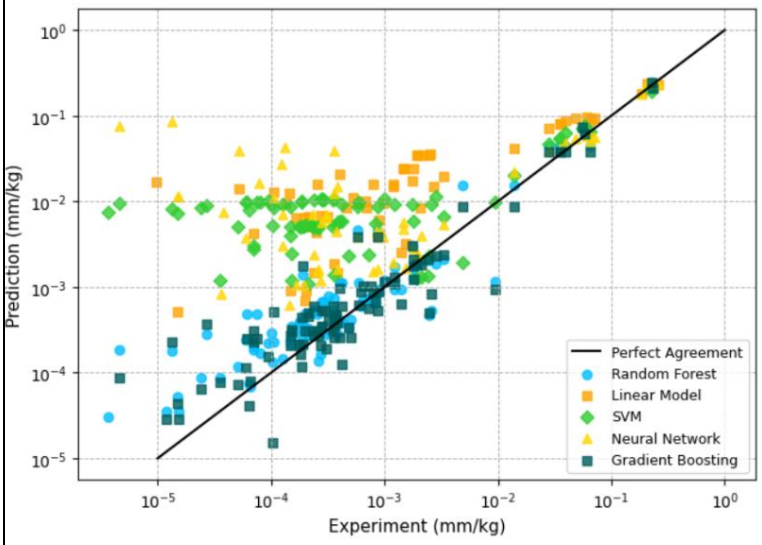

Figure 7. Results Of Random Split (RF,SVM, GB) And Stratify Split (NN,LR)

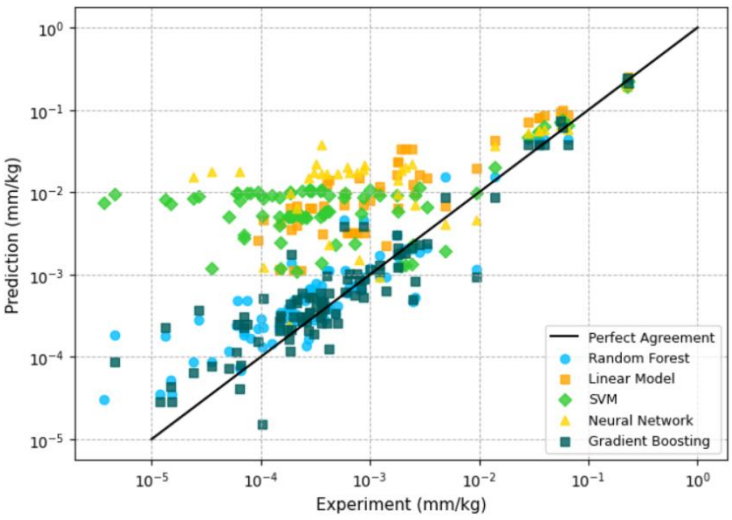

Figure 8. Results Of Random Split

The observed variation in performance may be due to how each model responds to the data distribution. Models like Linear Regression and Neural Networks might benefit from a balanced representation of different data regimes, while tree-based models and SVR may be more robust to such imbalances. However, it's also possible that the improved results under random splitting were simply due to a more favorable randomization in the train/test allocation. Further testing with multiple random seeds would be needed to isolate the true cause.

Conclusion

The models were applied to predict erosion rates using experimental data. Gradient Boosting achieved the highest accuracy among all models that trained with stratified splitting, while Random Forest performed best under a random split. Despite working with a small and partially inconsistent dataset, both models demonstrated strong predictive capabilities. However, the limited data size and inconsistencies in target labels constrained overall performance improvements. These results also highlight how the choice of data splitting strategy can influence model outcomes. With enhanced data quality and a larger sample size, these models have the potential to become reliable tools for industrial erosion monitoring and prediction. Although our models outperformed the article's baseline, further evaluations using additional metrics such as adjusted R^2 and cross-validation would provide a more reliable and comprehensive assessment.

References

[1] P. Zahedi, S. Parvandeh, A. Asgharpour, B. S. McLaury, S. A. Shirazi, and B. A. McKinney, "Random forest regression prediction of solid particle Erosion in elbows," *Powder Technology*, vol. 338, pp. 983–992, doi: https://doi.org/10.1016/j.powtec.2018.07.055

[2] Scikit-learn, "scikit-learn: Machine Learning in Python," *Scikit-learn.org*, [Online]. Available: https://scikit-learn.org/stable/ (accessed Jul. 5, 2025).

[3] Pandas, "General functions — pandas 1.4.2 documentation," *pandas.pydata.org*. [Online]. Available: https://pandas.pydata.org/docs/reference/general_functions.html (accessed Jul. 4, 2025).

[4] GeeksforGeeks, "Regression Metrics, Metrics for Machine Learning," *GeeksforGeeks*, [Online]. Available: https://www.geeksforgeeks.org/machine-learning/metrics-for-machine-learning-model/#:~:text=was%20actually%20Yes.-,Regression%20Metrics,-In%20the%20regression (accessed Jul. 8, 2025).