# Data Preprocessing and Modeling Report

## 1. Introduction

This report summarizes the data preprocessing steps and modeling outcomes applied to a capacitance dataset. The goal is to clean, transform, and prepare the dataset for regression modeling, evaluating the effect of different preprocessing pipelines.

## 2. Data Loading and Initial Cleaning

- Dataset loaded from `regression\_benchmark/data/capacitance\_data.csv`.  
- The 'DOI' column was dropped.  
- Empty strings in the 'Materials-2' column were replaced with NaNs, then filled with a placeholder '\_\_MISSING\_\_'.  
- An OrdinalEncoder was used to encode 'Materials-2', with '\_\_MISSING\_\_' treated as a distinct category.  
- An OrdinalEncoder was also used for ‘Materials-1’  
- Rows with Missing values were dropped

## 3. Exploratory Analysis

- A pairwise plot was generated to examine correlations with the target variable 'Capacitance (F/g)'. This column was first divided into 3 buckets to encapsulate range relationships  
A screenshot of a graph

AI-generated content may be incorrect.

- Strong linear correlations were noted with 'Specific surface area (m2/g)', 'Pore size (nm)', and 'N (at. %)', with 'Specific surface area' being the strongest. This was further supported by a heatmap.  
A diagram of a capacitors

AI-generated content may be incorrect.

- Pairwise plot between the strongest linear correlations with the target column can be seen below:   
A group of graphs showing different sizes of data

AI-generated content may be incorrect.

## 4. Outlier Detection and Handling

- 'Voltage window (V)' and 'Id/Ig' showed high positive skew and kurtosis, indicating potential outliers.  
- Log transformations were applied to these features followed by z-score normalization.  
- Three versions of the dataset were retained: raw dataset without any normalization normalized dataset, and lastly a normalized dataset with outliers removed

## 5. Data Splitting

- The dataset was split into training and testing sets.  
- Copies of the raw, outlier-included, and outlier-removed datasets were saved.  
- Raw dataset: train\_raw.csv, test\_raw.csv  
- Normalized dataset: train\_scaled.csv, test\_scaled.csv  
- Normalized dataset with outliers removed: train\_scaled\_removed\_outliers.csv, test\_scaled\_removed\_outliers.csv  
- These files can be found in the data folder

## 6. End Goals

- The primary objective is to compare the performance of models trained on the raw dataset and normalized datasets with and without outliers.  
- The effectiveness of different preprocessing strategies is assessed to determine the optimal pipeline for this data type.

# Model Training and Result Logging Report

## 1. Models Used

The project leverages a comprehensive set of regression models organized in a registry. These models include linear models, ensemble methods, neural networks, and gradient boosting variants:

- \*\*Linear Models\*\*: Ridge, Lasso, Bayesian Ridge, ElasticNet  
- \*\*Tree-Based Models\*\*: Decision Tree, Random Forest, Extra Trees  
- \*\*Boosting Models\*\*: Gradient Boosting, AdaBoost, HistGradientBoosting  
- \*\*Support Vector Regression\*\*: SVR  
- \*\*K-Nearest Neighbors\*\*: KNN  
- \*\*Neural Networks\*\*: MLPRegressor  
- \*\*Gradient Boosting Frameworks\*\*: XGBoost, LightGBM (commented out), CatBoost  
- \*\*Ensemble Models\*\*: Various StackingRegressors with different meta-models (Ridge, Lasso, BayesianRidge, MLP, ElasticNet, CatBoost)

## 2. Training and Evaluation

The `train\_all\_models` function performs model training and evaluation using GridSearchCV with specified hyperparameter grids. Each model is evaluated using the following metrics on the test data:

- R² score  
- Root Mean Squared Error (RMSE)  
- Mean Absolute Error (MAE)  
- Mean Absolute Percentage Error (MAPE)  
- Normalized RMSE (RMSE / target range)

Each model's best hyperparameters and cross-validation scores are also logged.

## 3. Saved Information Per Model

The following details are saved per model run in the results object:

- `model`: Model name  
- `success`: Whether the model training succeeded  
- `score\_mean\_cv`: Best cross-validation score  
- `best\_params`: Best hyperparameters  
- `estimator`: Best model instance  
- `r2\_test`, `rmse\_test`, `mae\_test`, `mape\_test`, `normalized\_rmse\_test`: Test set performance metrics  
- `error`: Exception message if training failed

## 4. Feature Importance Analysis

SHAP (SHapley Additive exPlanations) is used to compute feature importance for each successful model. A unified DataFrame is generated with the following columns:

- `Feature`: Input feature name  
- `Mean Importance`: Average absolute SHAP value across test samples  
- `Model`: Model name

## 5. Summary Generation

The `generate\_summary` function compiles all results into a pandas DataFrame containing key model metrics and hyperparameters, sorted by descending Test RMSE score.

## 6. Interpreting Key Files

The following files are generated from various model training experiments. These are categorized by the type of preprocessing applied (raw, scaled, or scaled with outliers removed), and whether direct models, ensemble models or v1 ensemble models were used.

- \*\*`\*\_model\_summary\_\*.csv`\*\*: Contains the evaluation metrics and hyperparameters for individual models. Each row represents a model, with columns for success status, CV R², test R², RMSE, MAE, MAPE, normalized RMSE, and best hyperparameters.

- \*\*`\*\_feature\_importance\_\*.csv`\*\*: Lists the mean absolute SHAP values for each feature per model. These help interpret which features were most influential in the model's predictions.

- \*\*`ensemble\_\*` and `v1\_ensemble\_\*` files\*\*: Represent results from more complex stacking ensembles. They include both model summaries and feature importance specific to ensemble configurations defined in `registry.py`.

- \*\*`saved\_models/`\*\*: A directory where serialized versions of the trained models (e.g., using pickle) are stored for future inference or analysis.

- \*\*`v1\_ensemble/`\*\*: A subdirectory specifically containing feature importance and summary files from the 'v1' ensemble configuration runs. Mirrors the structure and content of the standard ensemble files.

Key files generated (can be found in the results folder):

* ensemble\_feature\_importance\_raw.csv
* ensemble\_feature\_importance\_scaled.csv
* ensemble\_feature\_importance\_scaled\_removed\_outliers.csv
* ensemble\_model\_summary\_raw.csv
* ensemble\_model\_summary\_scaled.csv
* ensemble\_model\_summary\_scaled\_removed\_outliers.csv
* feature\_importance\_raw.csv
* feature\_importance\_scaled.csv
* feature\_importance\_scaled\_removed\_outliers.csv
* model\_summary\_raw.csv
* model\_summary\_scaled.csv
* model\_summary\_scaled\_removed\_outliers.csv
* saved\_models
* v1\_ensemble/
  + v1\_ensemble\_feature\_importance\_raw.csv
  + v1\_ensemble\_feature\_importance\_scaled.csv
  + v1\_ensemble\_feature\_importance\_scaled\_removed\_outliers.csv
  + v1\_ensemble\_model\_summary\_raw.csv
  + v1\_ensemble\_model\_summary\_scaled.csv
  + v1\_ensemble\_model\_summary\_scaled\_removed\_outliers.csv