SVM-based Classification and Bioelectrochemical Performance Prediction

Table of Contents

Abstract

This report provides a comprehensive mathematical and algorithmic explanation of the ML project on the classification of Bio-reactors. The project leverages experimental data from a controlled study on Microbial ElectroSynthesis (**MES**) to build a Support Vector Machine (**SVM**) model to classify the reactors based on their **performance** metrics, in the presence of **Nickel doping**. The main objective of the project is to train an SVM model, hyperparameter optimization/tuning, and interpretation.

Problem Statement

- The performance of bioelectrochemical reactors, particularly in the context of Microbial Electrosynthesis (**MES**), is governed by a complex interplay of electrochemical, biochemical, and environmental factors.
- The **manual classification** of bio-reactor performance, based on input parameters, is a tedious process. Conventionally, assessing and classifying a reactor's efficiency involves manual analysis of various parameters like product concentration, electron recovery rates, and biomass density, etc…
- There is a need for **novel**, **automated** methods to classify reactor data based on performance.

a. Objective

- The primary goal of this project is to develop a classification model to accurately predict reactor type based on its performance.
- The following are key objectives:
    1. **Data engineering:** To parse, clean and merge data from various sources into a single master dataset.
    2. **Exploratory Data Analysis (EDA):** To understand the patterns, correlations, and distributions within the data.
    3. **Model Development:** To implement an SVM classifier to study non-linear patterns/problems.
    4. **Optimization/Tuning:** To tune model hyperparameters using 'GridSearchCV' and cross-validation to maximise prediction accuracy.
    5. **Evaluation:** To evaluate the performance of the model through metrics (like f1-score, accuracy, precision, etc…), and to analyze feature importance.

Scientific Background

a. Microbial Electrosynthesis (MES)

- Microbial Electrosynthesis (MES) is a cutting-edge bioelectrochemical technology that utilizes microorganisms as bio-catalysts to convert simple, low-energy molecules into more complex, valuable chemicals.
- It is a "sustainable bioelectrochemical process that uses microorganisms to convert CO_2 into valuable chemicals by leveraging electrical energy."
- The overall process can be summarised as follows:

$$CO_2 + H_2O + e^- \xrightarrow{Microorganisms} Acetate + Formate$$

- In a typical MES system, a bio-reactor contains an anode and a cathode submerged in an electrolyte. At the cathode, electroactive microorganisms accept electrons supplied from an external power source. These electrons are then used to reduce a substrate, most commonly carbon dioxide (CO_2), into organic compounds.
- The efficiency of this conversion depends on:
  - Reactor design
  - Microbial Consortium
  - Operating conditions (pH, temperature)
  - Material of Cathode (it must transfer electrons efficiently and attach biofilm).

b. Role of Nickel (Ni) doping in Biocathodes

- The experimental dataset for this project investigates the effect of doping an activated carbon (AC) cathode with nickel (Ni). Activated carbon is a common cathode material due to its high surface area, good conductivity, and biocompatibility.
- However, its catalytic activity for CO_2 reduction can be limited. Nickel is a known electrocatalyst for various reactions, including the hydrogen evolution reaction (HER) and CO_2 reduction.
- In the context of MES, doping the AC cathode with Ni nanoparticles is hypothesized to enhance performance through several mechanisms:
  - **Improved Electron Transfer:** Ni particles can act as electron conduits, improving the electrical connection between the cathode surface and the microbial cells.
  - **Catalytic Activity:** Nickel itself can catalyze the reduction of CO_2 or intermediate compounds, potentially lowering the energy barrier for the synthesis of products like acetate.

  - **Hydrogen Production**: Nickel can catalyze the production of H_2, which can then be used by some microorganisms as an electron donor for CO_2 reduction.
- The study explores this by comparing reactors with different levels of Ni doping: 0% (Control), 0.01% (Low-Ni), and 5% (High-Ni). The model's ability to distinguish reactors shows the influence of Nickel.

Dataset Overview

a. Data Source

The dataset is an MS Excel spreadsheet with Microbial electrosynthesis (MES) experimental data from bio-reactor studies testing nickel-doped activated carbon (AC) cathodes for CO2 conversion to acetate.

- **Header Column:** Experimental Time in Days

$$Header \rightarrow Row_0(Reactor\,Label)+Row_1(Parameter\,Description)$$

- **Data in each sheet and their corresponding parameters:**

Sheet	Column 1	Column 2	Purpose

IVIUM 16.5/24.5/Combined	Time (days)	time/s;
		I/mA;
		Q/mC (repeated per reactor)
			Electrochemical measurements

R1-R6	Date	Time (days)	Individual reactor performance: volumes, formate/acetate concentrations, charges, electron recovery
OD	Day	OD measurements + std dev (per reactor type)	Planktonic biomass growth tracking
OD with Dilution	Day	Control; AC-Ni 0.01%; AC-Ni 5% (each with std)	Dilution-corrected growth data

NH4	-	NH4 (mg/L) × 8 columns	Ammonium concentration
pH	-	pH × 8 columns	pH values
Data for Graphs	Date	Time (days)	Consolidated formate, acetate, e-recovery data organized for plotting
Data Abiot H2	Day	AC control; AC-Ni 0.01%; AC-Ni 5% (duplicates)	Baseline H_2 production without microbes

Table 1 - Dataset Overview

b. Reactor Configurations

The six reactors are classified into three distinct categories based on the composition of the cathode material. This classification forms the target variable (***'y' variable)*** of the supervised learning model.

Reactor	Doping Category	Nickel %	y - value	Description
R_1, R_2	Control	0	0	Baseline reactors with standard Activated Carbon (AC) cathodes
R_3, R_4	Low Ni	0.01	1	Reactors with AC cathodes lightly doped with Nickel.
R_5, R_5	High Ni	5	2	Reactors with AC cathodes heavily doped with Nickel.

Table 2 - Reactor classification labels

c. Measured Parameters (Features)

The dataset contains a rich set of features that describe the state and performance of the reactors. These are the inputs (***'X' variable***) to the model.

Feature (X)	Symbol	Unit	Interpretation
Time	t	days	The temporal progression of the experiment. Used to track dynamic changes.
Current	I	mA	Electrochemical activity. A higher (negative) current indicates a faster rate of electron consumption by the microbes.
Charge	Q	mC	Cumulative electrochemical activity. The integral of current over time, representing the total electrons consumed.
Formate Concentration	C1	mg/L	Concentration of formate, a potential product of CO_2 reduction.
Acetate Concentration	C2	mg/L	Concentration of acetate, the primary target product of MES in this study. A key performance indicator.
Electron Recovery	e- recovery	%	The percentage of consumed electrons that are "recovered" in the form of a specific chemical product (e.g., C1 or C2). A measure of coulombic efficiency.
Optical Density	OD	a.u.	A proxy for biomass concentration. Higher OD suggests more microbial growth.
pH	pH	-	Environmental stability. Microbial activity is highly sensitive to pH.
Ammonium Concentration	NH4	mg/L	Concentration of a primary nitrogen source, essential for microbial growth.
Nickel Concentration	Ni Conc.	wt%	The engineered variable defining the reactor type. Explicitly included as a feature.

Table 3 - Features description

Technical Background

a. Python

- Python is highly favored for machine learning projects due to its extensive library ecosystem, including tools like NumPy, Pandas, Scikit-learn, TensorFlow, and PyTorch that speed up development and reduce the need for custom coding.
- Its readable and simple syntax makes it accessible and easy to maintain, allowing developers to focus on algorithm design rather than technical complexities.
- Python offers great flexibility, supporting both object-oriented and scripting styles, and it enables seamless integration with other languages and platforms. Its platform independence allows code to be run across different operating systems with minimal changes, saving time and effort during deployment.
- Furthermore, Python has a large, active community that fosters collaboration, innovation, and continuous improvement of machine learning tools and resources. This combination of features enhances efficiency, scalability, and facilitates rapid prototyping in machine learning projects.

b. Libraries Utilized

- **NumPy**: Fundamental package for numerical computing and array operations.
- **Pandas**: Data manipulation and analysis, handling structured data.
- **Matplotlib and Seaborn**: Visualization tools for plotting and interpreting data.
- **Scikit-learn**: Core library for classical machine learning algorithms including classification, regression, and clustering.

c. Supervised Learning

i. Definition

1. Supervised learning is a machine learning approach where models are trained using labeled datasets, meaning each input data point is paired with the correct output label.
2. The model learns to map inputs to outputs by minimizing errors during training, enabling accurate predictions on new, unseen data. It is widely used for tasks like classification, where outputs are discrete categories, and regression, where outputs are continuous values.
3. Supervised learning relies on quality data, iterative model refinement, and is fundamental in applications such as image recognition, medical diagnosis, and recommendation systems due to its ability to predict and classify based on past examples.

ii. Classification

1. Classification in machine learning is a supervised learning approach where the objective is to assign input data into predefined categorical classes or labels. It is widely used in applications such as spam detection, medical diagnosis, image recognition, and sentiment analysis.
2. Classification can be binary, where there are two classes, or multi-class, where data points belong to multiple discrete categories. The models learn from labeled training data to predict the class of new, unseen instances.
3. Common algorithms include decision trees, **support vector machines**, Bayes, and neural networks, each suited for different types of classification problems and data structures.

iii. Kernel Trick

1. The kernel trick is a technique that lets algorithms like SVMs learn nonlinear decision boundaries while still using a mathematically "linear" model.
2. Instead of explicitly mapping data into a high dimensional feature space, the algorithm uses a kernel function to compute inner products there implicitly.
3. This gives the benefits of richer feature spaces and nonlinear separation without the heavy computational cost of actually computing the transformed features.
4. The **Kernel Trick** is:

$$k(x_i, x_j) =< \phi(x_i), \phi(x_j) >= \phi(x_i)^T \phi(x_j)$$

iv. Support Vector Machine

1. Definition

a. Support Vector Machine (SVM) classification is a supervised learning algorithm that finds the optimal hyperplane to separate different classes in a dataset by maximizing the margin between the classes. SVM uses support vectors, which are the nearest data points to the hyperplane, to define this boundary.

b. The algorithm can handle both linear and non-linear classification by using the kernel trick, which implicitly maps data into higher-dimensional spaces to make them linearly separable.

c. SVM is robust to outliers due to margin maximization and is widely applied for classification due to its accuracy and generalization capabilities.

d. It works well in high-dimensional spaces and various practical applications.

2. SVM Formulation

a. For a given labelled data $(x_i, y_i)^n_{i=1}$, SVM solves the optimization problem:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

subject to:

$$y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i;\ \xi_i \geq 0;\ \forall i$$

b. **Parameters**:

i. w - weight vector (hyperplane normal)
ii. b - bias term
iii. ξ - slack variable
iv. C - regularization parameter
v. $\phi(.)$ - Kernel mapping function

3. Multi-Class Formulation

a. For K classes, the classification strategy is:

$$\binom{K}{2} = 3\ binary\ classifiers$$

b. The classifiers are:

i. Control 1: Control vs Low Ni
ii. Control 2: Low Ni vs High Ni
iii. Control 3: High Ni vs Control

4. RBF Kernel

a. **RBF** - Radial Basis Function

b. RBF Kernel function can be defined as:

$$k(x_i, x_j) = exp(-\gamma||x_i - x_j||_2^2)$$

where γ - kernel bandwidth

c. **Mathematical Interpretation:**

i. Small γ: Smooth decision boundary; underfitting risk.
ii. Large γ: Complex boundary; overfitting risk.
iii. RBF maps into infinite-dimensional space by Kernel Trick.

Model Implementation Algorithm

- The algorithm/process flowchart of the SVM model implementation is as follows:

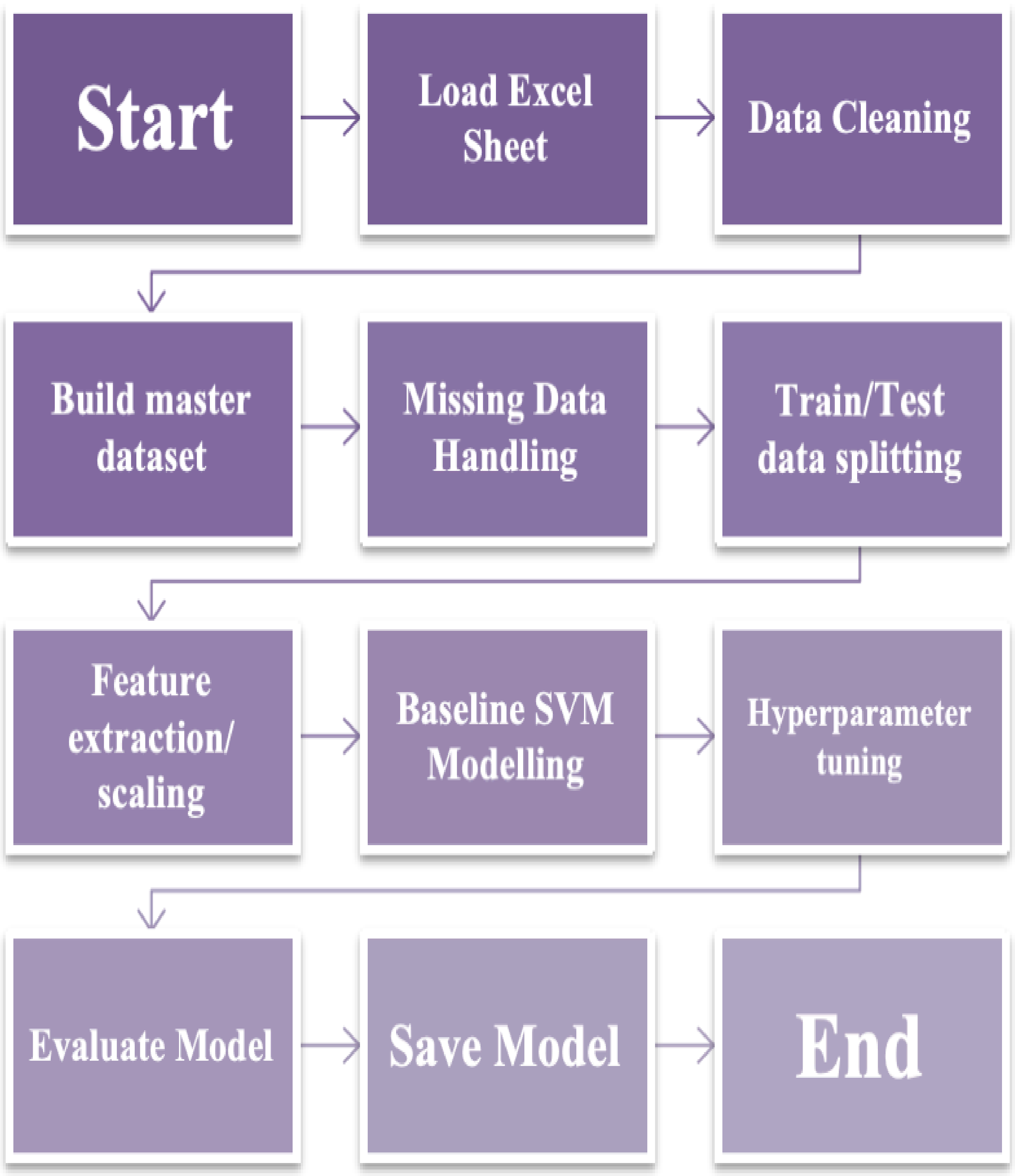


Fig. 1 - Algorithm Flowchart

Project Implementation

a. Start of the Project

- All the necessary files are gathered.
  - **Dataset** - Excel file with AC cathode data.
  - **Google Colab** - To implement Python code.

b. Data Loading

- Python's pandas module is used to load the data into the python script.

c. Data Cleaning

- The dataset contains many missing values, which can be eliminated/removed by converting them into NaN (not a number).
- This conversion helps in cleaning the data, as these rows are avoided in further operations.
- Numeric format:

$$x_{ij} \in \mathbb{R} \cup \{NaN\}$$

d. Master Dataset

- Master Dataset is the combination of all the features and its corresponding labels.
- **Uncleaned Master Dataset:**
  - **Reactors:** 6
  - **Days:** 48
  - **Features:** 14
    - **Electrochemical metrics**: Current (I/mA), Charge (Q/mC), Electron recovery (%)
    - **Biochemical products**: Formate (C1, mg/L), Acetate (C2, mg/L)
    - **Biological indicators**: Optical Density (OD), Ammonium (NH4, mg/L)
    - **Environmental parameters**: pH, reactor volume (ml)
  - **Total Samples:** 288 (48 samples x 6 reactors)

e. Data Handling

- Complete clean set is defined as:

$$C = \{i : \forall j \in [1, d], X_{ij} \neq NaN\}$$

- **Clean dataset Samples:** 126 (42 x 3 class)

f. *Train/Test Split*

i. Stratified K-fold Cross Validation

1. Stratified K-fold Cross-Validation is a variation of the K-fold cross-validation technique used for evaluating machine learning models, especially in classification problems where class labels are imbalanced. It ensures that each fold of the dataset maintains the same proportion of samples from each class as in the entire dataset.
2. This stratification ensures that every fold is a good representative of the whole, helping to produce more reliable and unbiased estimates of model performance.
3. Unlike basic K-fold cross-validation, which randomly splits data into folds, stratified K-fold pays special attention to preserving class distribution within each fold, thus avoiding situations where minority classes are underrepresented or missing in some folds.
4. This is particularly important in imbalanced datasets, like medical diagnosis data where one class may dominate the sample. The training and testing sets in each fold maintain the class ratios, leading to better generalization and more accurate performance metrics.

ii. Data Split

1. **Train:** 70%
2. **Test:** 30%

iii. Mathematical Interpretation

$$\text{Stratified Split} : \min_{\text{split}} \sum_{k=0}^{2} |p_k^{\text{train}} - p_k^{\text{full}}|$$

where p_k - proportion of class k in dataset.

iv. Class Balance

1. Given $n_0 = 42$, $n_1 = 42$, $n_2 = 42$ (balanced by design)

$$N_{train} = 0.7 * 126 = 88.2 \approx 88$$

$$N_{test} = 0.3 * 126 = 37.8 \approx 38$$

g. *Feature Engineering*

i. Feature Selection

1. Depending on the correlation between each feature, the dimension can be reduced depending on the highest correlation.
2. The key features are:

Feature	Symbol	Unit	Interpretation
Acetate Concentration	C_2	mg/L	Product yield
Electron recovery	η_e	%	Bioelectrochemical efficiency
Optical Density	OD	au	Biomass proxy
pH	pH	-	Environmental stability
Ni concentration	[Ni]	wt%	Catalyst loading
Experiment Day	t	day	Temporal progression

Table 4 - Key Features Selected

ii. Feature Matrix formulation

1. Given n clean samples and d features:

$$X \in \mathbb{R}^{n \times d}, \quad X_{ij} = (\text{i-th sample, j-th feature})$$

2. Target variable:

$$y \in \{0, 1, 2\}^n$$

iii. Exploratory Data Analysis

1. Definition

a. Exploratory Data Analysis (EDA) is a crucial data science process that involves analyzing and visualizing datasets to understand their main characteristics.
b. It helps identify patterns, spot anomalies, detect relationships, and test hypotheses, ensuring data quality and preparing data for further analysis or modeling.

c. EDA supports better decision-making and model accuracy by revealing insights that guide subsequent steps in data analysis.

2. Mathematical Model

a. Descriptive Statistics

i. For each feature *f*:

$$\mu_f = \frac{1}{n}\sum_{i=1}^{n} x_{if}, \quad \sigma_f = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{if} - \mu_f)^2}$$

b. Distribution Analysis

i. Box plot

ii. Time Series Analysis

c. Class Separability

i. For linear separability assessment, between-class variance:

$$\Sigma_B = \sum_{k=0}^{2} n_k(\mu_k - \mu_{\text{global}})(\mu_k - \mu_{\text{global}})^T$$

ii. where,

1. n_k - samples in class k
2. μ_k - class mean
3. μ_k - global mean

iii. For non-linear assessment - **RBF**

3. Results

a. Acetate Production

i. This plot shows the high dependence of Product Yield on Ni Doping Level.

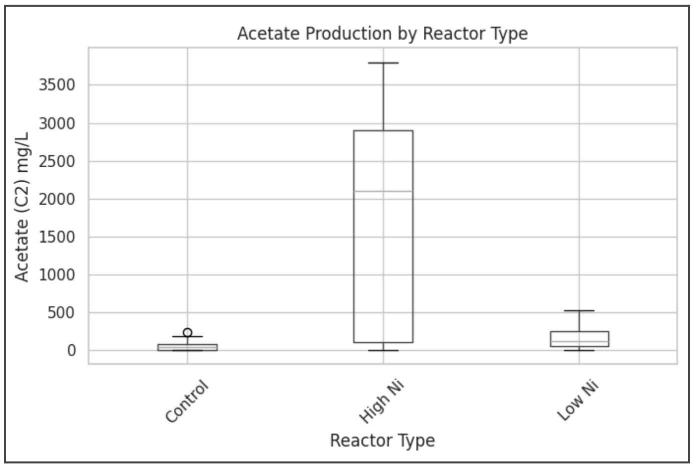


Fig. 2: Acetate Production by Reactor Type

b. Electron Recovery

i. This plot shows the electron recovery of each reactor category (0, 1, 2) over time.

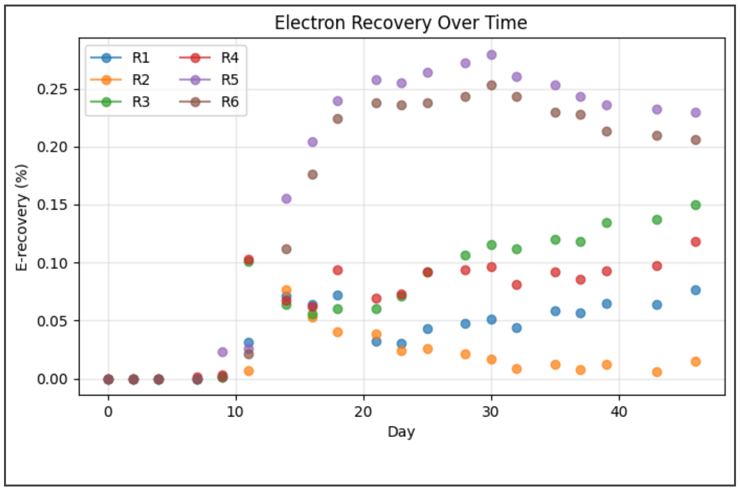


Fig. 3: e- recovery over time

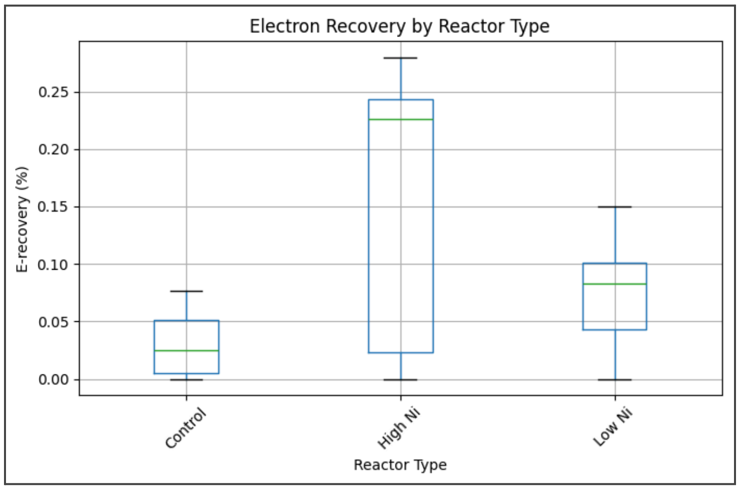


Fig. 4: e- recovery by Reactor Type

iv. Feature Scaling

1. Data Standardisation

a. StandardScalar Transformation

i. Before SVM training, features are standardized using the StandardScaler from scikit-learn:

$$\tilde{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

ii. where:

$$\mu_j = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} x_{ij}^{\text{train}}$$

$$\sigma_j^2 = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} (x_{ij}^{\text{train}} - \mu_j)^2$$

iii. SVM optimization is case-sensitive. Unscaled features with larger magnitudes dominate the kernel computation, leading to biased decision boundaries.

iv. Standardization ensures equal feature importance in the feature space.

b. Scalar Fitting and Transformation

i. Fit: Compute $\boldsymbol{\mu_j}$, $\boldsymbol{\sigma_j}^2$ from training data only.

ii. Transform Train: Apply standardisation to X_{train}.

iii. Transform Test: Apply standardisation to X_{test}.

2. Feature Importance

a. Permutation Importance

i. For a feature f, randomly shuffle its values and measure accuracy drop:

$$I_f = A_{\text{orig}} - A_{\text{shuffled}}$$

ii. Computed as average over multiple random permutations:

$$I_f = \frac{1}{m} \sum_{\ell=1}^{m} (A_{\text{orig}} - A_{\text{shuffled},\ell})$$

iii. where, m = 10 permutations in this implementation.

iv. Mathematical Interpretation:

1. ***$I_f > 0$***: Permutating reduces model performance; positive contribution to model prediction performance.
2. ***$I_f \simeq 0$:*** Permutating does not affect model performance.
3. ***$I_f < 0$:*** The model gets affected, i.e. the model may get noise.

v. Sci-kit Learn Implementation

```Python
perm_imp = permutation_importance (model, X_test, y_test, n_repeats=10)
```

b. Cross–Validation

i. Divide dataset into K = 3 classes, each maintaining class proportions:

$$\text{CV-Error} = \frac{1}{k}\sum_{i=1}^{k} L(M_i, X_{\text{val}}^{(i)}, y_{\text{val}}^{(i)})$$

ii. Where, L - loss function (e.g. classification error, f1-score).

c. Validation Estimation

i. Standard error of cross-validation:

$$\text{SE(CV-Error)} = \sqrt{\frac{1}{k(k-1)}\sum_{i=1}^{k}(\text{Error}_i - \overline{\text{Error}})^2}$$

ii. **Confidence interval:**

$$\text{CV-Error} \pm 1.96 \times \text{SE}$$

3. Performance Metrics

a. Confusion Matrix

i. For multi-class classification, define:

$$C \in \mathbb{Z}^{3\times 3}, \quad C_{ij} = \#\{n : y_n = i, \hat{y}_n = j\}$$

where, *y_cap* - predicted labels.

b. Multi-Class

i. Accuracy

$$\text{Acc} = \frac{\sum_{i=0}^{2} C_{ii}}{\sum_{i,j} C_{ij}}$$

ii. Precision

$$P_k = \frac{C_{kk}}{\sum_i C_{ik}}$$

iii. Recall

$$R_k = \frac{C_{kk}}{\sum_j C_{kj}}$$

iv. F1–score (macro weighted)

$$F_1^{(k)} = 2 \cdot \frac{P_k \cdot R_k}{P_k + R_k}$$

$$F_1^{(\text{weighted})} = \frac{1}{n}\sum_{k=0}^{2} |y_k| \cdot F_1^{(k)}$$

4. Error Metrics

a. Generalisation Error:

$$\widehat{E}_{\text{gen}} = \frac{1}{k}\sum_{i=1}^{k} L_i$$

b. Standard Error:

$$\text{SE}(\widehat{E}_{\text{gen}}) = \sqrt{\frac{\sum_{i=1}^{k}(L_i - \overline{L})^2}{k(k-1)}}$$

c. Confidence interval (95% accuracy):

$$\text{Acc} \pm 1.96 \times \sqrt{\frac{\text{Acc}(1-\text{Acc})}{n_{\text{test}}}}$$

h. Baseline SVM Classifier (SVC) Model

i. Parameter Values (Default)

1. Scikit-Learn has a default set of values:
   a. C = 1.0
   b. $\gamma = \frac{1}{n_{features}.Var(x)}$

ii. Results

1. Confusion Matrix

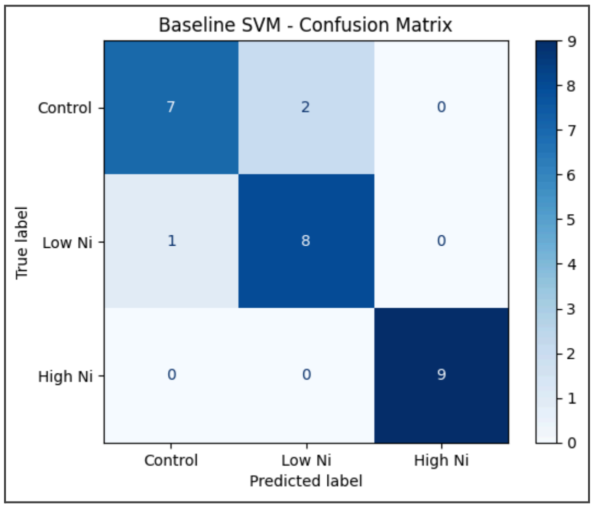
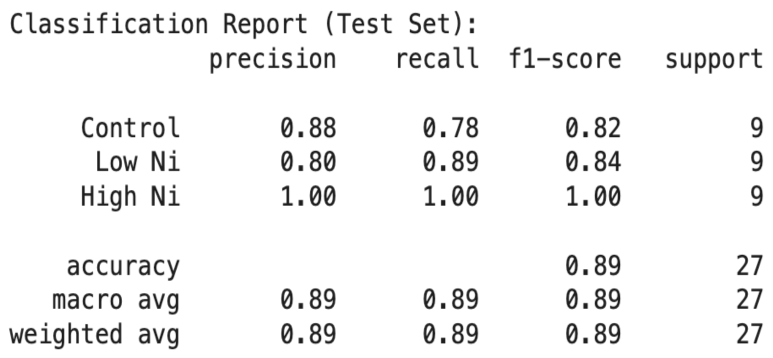


Fig. 5: Confusion Matrix (Baseline - C=1; gamma=scale/auto)

2. **Classification Report**



```
Classification Report (Test Set):
              precision    recall  f1-score   support

     Control       0.88      0.78      0.82         9
      Low Ni       0.80      0.89      0.84         9
     High Ni       1.00      1.00      1.00         9

    accuracy                           0.89        27
   macro avg       0.89      0.89      0.89        27
weighted avg       0.89      0.89      0.89        27
```

3. **Scores**

a. Training Accuracy:	**0.8730**
b. Test Accuracy:	**0.8889**

i. Hyperparameter Tuning

i. GridSearchCV

1. **Definition**
    a. GridSearchCV is a machine learning method from scikit-learn used for hyperparameter tuning.
    b. It performs an exhaustive search over a specified grid of hyperparameter values for a given model, systematically training and evaluating the model with each combination using cross-validation.
    c. This process helps identify the best hyperparameters that optimize model performance according to a chosen scoring metric.
    d. GridSearchCV takes as inputs the estimator (model), the parameter grid to search, the cross-validation strategy, and the scoring metric, and returns the best combination of parameters along with the best model fit.

2. **Evaluation Metrics**
    a. GridSearchCV uses the following set of parameters for the SVM model:
        i. $\mathcal{G}$ = {0.1, 1, 10, 100} x {0.001, 0.01, 0.1, ‘auto’, ‘scale’}
        ii. (i.e.) $\boldsymbol{\mathcal{G}}$ **= C x** $\boldsymbol{\gamma}$

b. For each {C, γ} pair, GridSearchCV evaluates generalization error via cross-validation to determine the best possible combination of parameters:

$$\text{CV-Score} = \frac{1}{k}\sum_{i=1}^{k} F_1(M_{-i}, X_{\text{val}}^{(i)}, y_{\text{val}}^{(i)})$$

Where,

1. M_{-i} - model trained on all folds except i;
2. X^{i}_{val} - validation set for fold i

3. Complexity

$$|\mathcal{C}| \times |\Gamma| \times k = 4 \times 5 \times 3 = 60 \text{ SVM trainings}$$

ii. Results - Optimized SVC

1. Best Parameters

a. C = **10**

b. γ = **0.1**

c. Kernel: **rbf**

d. CV F1 - score: **0.7857**

2. Confusion matrix

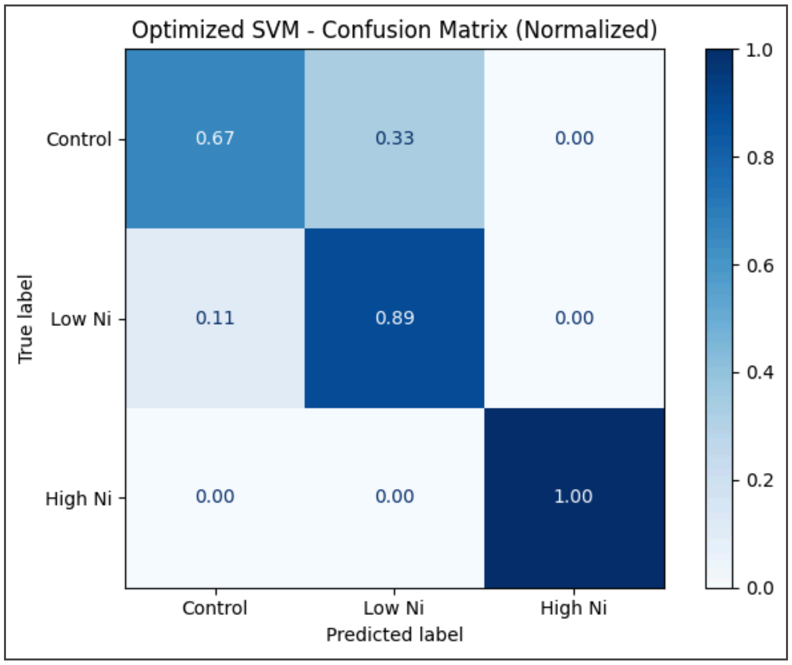
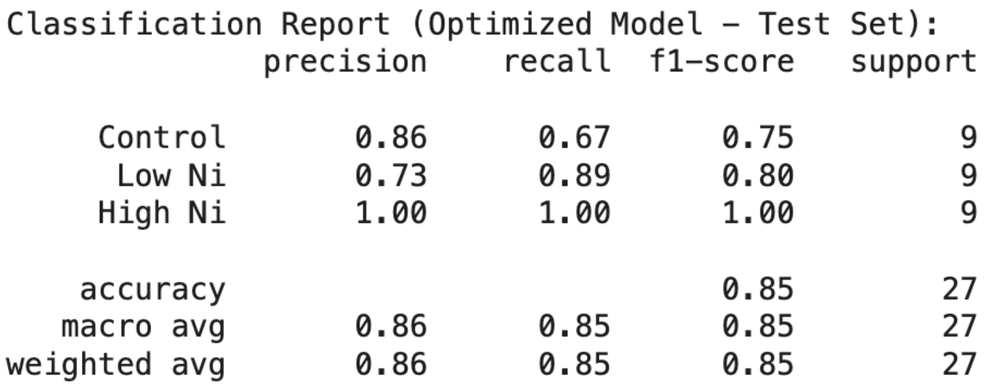


Fig. 6: Confusion matrix (Optimized+Normalized)

3. Classification Report



```
Classification Report (Optimized Model - Test Set):
              precision    recall  f1-score   support

     Control       0.86      0.67      0.75         9
      Low Ni       0.73      0.89      0.80         9
     High Ni       1.00      1.00      1.00         9

    accuracy                           0.85        27
   macro avg       0.86      0.85      0.85        27
weighted avg       0.86      0.85      0.85        27
```

4. Scores

a. Training accuracy: **0.8889**
b. Test accuracy: **0.8519**
c. Test F1-score (weighted): **0.8500**

Model Evaluation - Results

a. Conclusion

i. The model achieved a high accuracy score of **~85%,** which helps in determining the performance of the model.

b. Insights on Feature Importance

i. The **Permutation Importance** metrics provides insights on the importance and weightage of each feature (i.e. highest correlation with the output).

ii. The Importance of each (selected) feature is as follows:

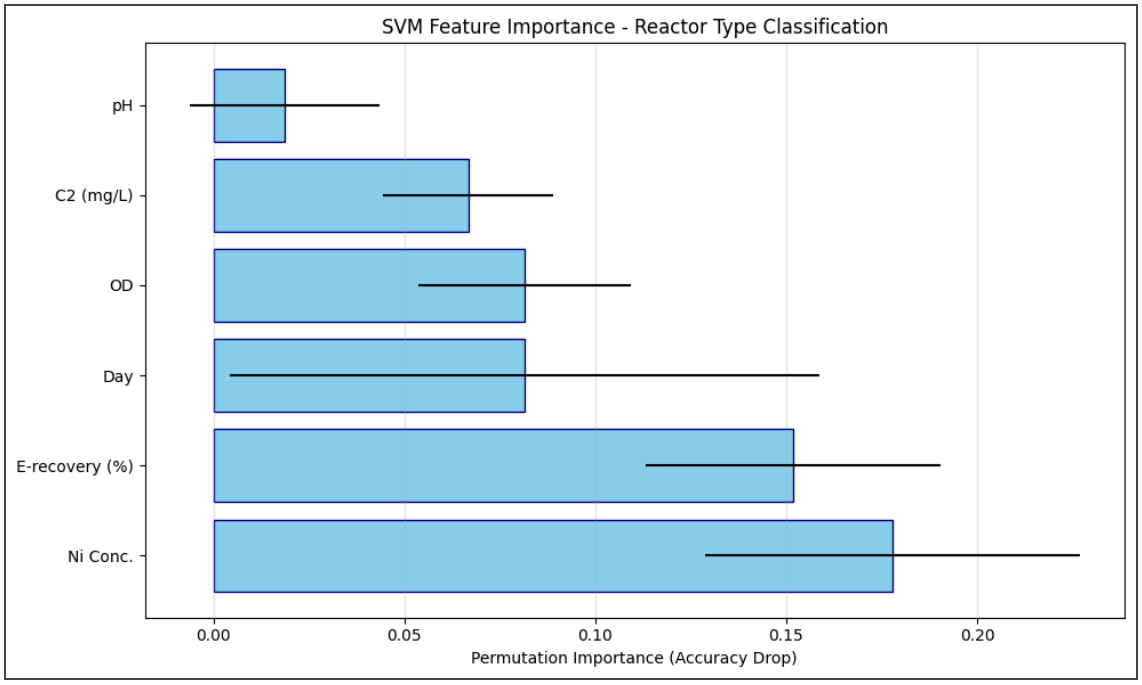


Fig. 7: Permutation Importance

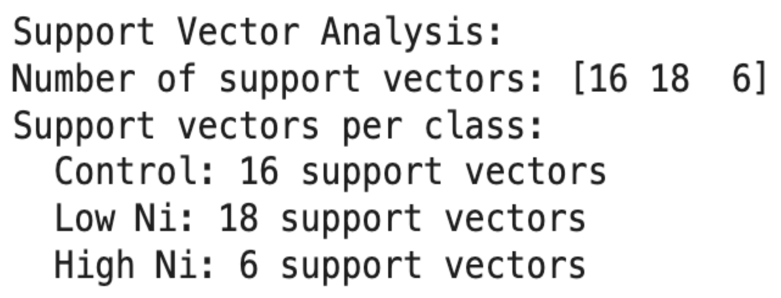
iii. The boxplot tells the weightage of each feature.

iv. Insights:

1. The model is heavily dependent on the ***Ni Concentration.***
2. The model is least dependent on the ***pH.***

c. *Final Insights*

i. Classification Category



```
Support Vector Analysis:
Number of support vectors: [16 18  6]
Support vectors per class:
  Control: 16 support vectors
  Low Ni: 18 support vectors
  High Ni: 6 support vectors
```

ii. Model Performance

1. Baseline SVM Model

a. **Parameters:**

   i. C = 1.0

   ii. $\gamma = \frac{1}{n_{features} . Var(x)}$

b. **Scores:**

   i. Training Accuracy: **0.8730**

   ii. Test Accuracy: **0.8889**

2. Optimized SVM Model

a. **Parameters:**

   i. C = 10.0

   ii. $\gamma =$ 0.10

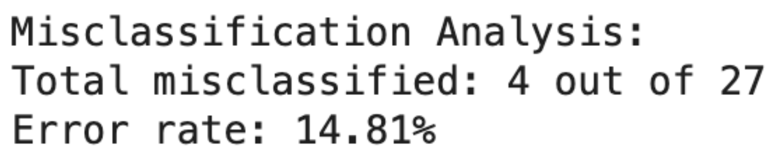
b. **Scores:**

   i. Training accuracy: **0.8889**

   ii. Test accuracy: **0.8519**

   iii. Test F1-score (weighted): **0.8500**

iii. Error Analysis



```
Misclassification Analysis:
Total misclassified: 4 out of 27
Error rate: 14.81%
```

References

a. Vapnik, V., & Cortes, C. (1995). "Support-vector networks."Machine Learning, 20(3), 273-297. (The foundational paper on Support Vector Machines).
b. 2. Schölkopf, B., & Smola, A. J. (2002).Learning with kernels: Support vector machines, regularization, optimization, and beyond. MIT press. (A comprehensive text on kernel methods).
c. 3. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). "Scikit-learn: Machine learning in Python."The Journal of Machine Learning Research, 12, 2825-2830. (The official paper for the scikit-learn library).
d. 4. Kohavi, R. (1995). "A study of cross-validation and bootstrap for accuracy estimation and model selection." InIJCAI, 14(1), 1137-1145. (A key paper on the importance and methodology of cross-validation).
e. 5. Breiman, L. (2001). "Random forests."Machine learning, 45(1), 5-32. (Introduced concepts related to permutation importance).
f. 6. Strobl, C., Boulesteix, A. L., Zeileis, A., & Hothorn, T. (2008). "Bias in random forest variable importance measures."BMC bioinformatics, 8(1), 25. (A detailed analysis of permutation importance).
g. 7. Bergstra, J., & Bengio, Y. (2012). "Random search for hyper-parameter optimization."The Journal of Machine Learning Research, 13(1), 281-305. (Compares grid search to random search for hyperparameter tuning).

Supplementary Materials

a. Project Code Script

i. MLPE-Project
ii. For Best Execution - Google Colab
iii. **Input -** Project Dataset in 'xlsx' format.

b. Project Dataset

i. AC-Ni-bio-reactor-dataset
ii. For Best Visuals – MS Excel

c. Project Master Dataset

i. Uncleaned Data (Master df) - MLPE
ii. For Best Visuals – Google Sheets

d. Project PPT

i. Project-MLPE
ii. For Best Visuals – Google Slides