Pattern Recognition Algorithms To Extract Meaningful Information From The Raw Data Produced By The Biosensor

A B.Tech Project report submitted by

Megavath Pavan (B21EE036)

and

Pujari Bheemesh (B21EE053)

in partial fulfilment of the requirements for the award of the degree of

B.Tech

INDIAN INSTITUTE OF TECHNOLOGY JODHPUR

DEPARTMENT OF ELECTRICAL ENGINEERING

18 November 2024

Declaration

I hereby declare that the work presented in this Project Report titled "*Pattern Recognition Algorithms to Extract Meaningful Information from the raw data produced by the Biosensor*" submitted to the *Indian Institute of Technology Jodhpur* in fulfilment of the requirements for the *B.Tech Project (EED4010)* is a bonafide record of the research work carried out under the supervision of *Dr Akshay Moudgil*. The contents of this Project Report in full or in parts, have not been submitted to, and will not be submitted by me to, any other Institute or University in India or abroad for the award of any course or project.

Signature

Megavath Pavan (B21EE036)

Pujari Bheemesh (B21EE053)

Acknowledgement

We are immensely grateful to our Teaching Assistant, *Ms Sukanya pandey (P22ID002)*, whose unwavering support and availability greatly enhanced our progress. They provided detailed explanations, technical assistance, and practical suggestions at every stage, ensuring that we had the resources and understanding necessary to overcome technical obstacles. Their hands-on approach and readiness to assist were invaluable, allowing us to delve deeper into the intricacies of wavelet-based attention mechanisms and multi-scale image compression. Their guidance was essential in helping us achieve our project goals.

Certificate

This is to certify that the Project Report titled "*Pattern Recognition Algorithms to Extract Meaningful Information from the raw data produced by the Biosensor*" submitted by *Megavath Pavan (B21EE036)* and *Pujari Bheemesh (B21EE053)* to the *Department of Electrical Engineering, Indian Institute of Technology Jodhpur* for the *B.Tech Project (EED4010)* is a bonafide record of the research work done by him under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any course or project.

Signature

Dr.Akshay Moudgil

Assistant professor

Dept. of Electrical Engineering

Indian Institute of Technology Jodhpur

Abstract

In this Project, We present the application of Advanced Machine Learning approaches for enhancing the analysis of raw data produced by biosensors. Biosensors are extensively used in detecting biological analytes and produce complex raw data that requires extensive signal processing techniques to extract meaningful information. Traditional methods of signal processing, while effective, often governed by fixed, rule-based logic that may not adapt to dynamic changes in behaviour of sensors and environment, struggle to handle the data with complex, nonlinear relations. To overcome these difficulties, we enhance or replace conventional signal processing techniques with Machine Learning Techniques and explore how they perform. This project integrates ML techniques at key stages of the Signal processing pipeline, including Noise reduction, Feature extraction, Pattern recognition and Decision Making. We check the reliability of the ML algorithms by testing these on complex, noised data, achieve adaptive noise filtering, automated feature extraction and real time pattern classification that lead to improved accuracy.

Table of Contents

1.Introduction

Biosensors are crucial devices in modern science and industry applications, enabling detection and analysis of various biological substances in medical diagnostics, drug screening, environmental monitoring, bioresearch and food safety. Biosensors are equipped with advanced nanotechnology, signal amplification techniques. However, noise is unavoidable in signals. Since most of the biosensors depend on bioreceptors that destabilise the performance of biosensors in various environmental conditions[4], we need accurate, adaptive and reliable signal processing techniques [1]. The implementation of Biosensor Signal processing unit with Advanced Machine Learning algorithms offers several advantages, including ability to adapt to new data, autonomously self-calibrate and handle complex, nonlinear signal patterns that traditional signal processing techniques struggle with [3]. There is no need to worry about Digital signal processing units, Fourier transforms, deriving complex mathematical models for raw data of Biosensor. ML algorithms have the potential to revolutionise biosensor technology [2], enhance or even replace conventional signal processing techniques, making it more reliable, efficient and scalable for healthcare, environmental monitoring [3], [5], diagnostics, point of care testing, future trend prediction and recommendation applications. Biosensors equipped with ML based Signal Processing can be deployed in real-time, implementable, wearable devices.

In general, The signal produced by biosensor in response to biological analyte concentration in the sample is analysed and meaningful information about the presence and concentration of the analyte is extracted. However, The raw data contains noise and complex patterns that require further processing. Signal processing methods like Noise Filtering, Feature Extraction, Signal Interpretation are used to arrive at a decision. Traditional approaches use fixed algorithms, rule based logic, which may not be adaptive enough when environmental conditions and sensor behaviour changes in real-time. To address these challenges, in this project, we have integrated Machine Learning techniques in the signal processing pipeline that focuses on stages such as noise filtering, feature extraction, Pattern recognition and decision making and demonstrated the potential for improved performance, real-time analysis and automated calibration ultimately resulting in more efficient, accurate, adaptive, robust and scalable solutions. In this project, For Noise Filtering: Wavelet transform, Addressing Class Imbalance: SMOTE(synthetic minority oversampling technique), Feature Scaling: Standardisation, Statistical Analysis: Mean, Median, Standard deviation, Feature Extraction: PCA, Decision making: Linear Regression, Lasso Regression, Support Vector Regression(SVR), Gaussian Process Regression,

Decision Tree Regression(DTR), Random Forest Regression(RFR), Gradient Boost Regression(GBR), K-nearest neighbours(KNN), K-means, Time Series K-means [6-7], DBSCAN, Hierarchical Clustering, Gaussian Mixture Model(GMM) and Long-Short Term Memory(LSTM) are used.

2.Problem Statement & Objective

As traditional signal processing techniques face challenges in handling noisy, complex data and struggle to adapt to various environmental conditions, Integrating machine learning techniques in signal processing units offers a solution by enabling adaptive noise reduction, automated feature extraction and continuous optimization. This project aims to enhance the biosensor performance through ML driven processing for improved accuracy and reliability.

The Primary objective of this project is to develop an enhanced signal processing approach for biosensor by integrating machine learning algorithms to improve the accuracy, adaptability and realtime processing of biosensor. Specifically we aim to:

1. Identify the optimal parameter values for the improved signal clarity and operation of a biosensor using its transfer characteristics.
2. Implement machine learning techniques for adaptive noise reduction to handle complex data and variable environmental conditions.
3. Automate feature extraction process using advanced methods such as Principal component analysis (PCA) that captures complex signal characteristics.
4. Apply machine learning for robust sensor calibration prediction, pattern recognition and classification to increase the reliability of analyte detection.
5. Evaluate the performance of proposed machine learning techniques through validating the results.

This approach is implemented to demonstrate how the integration of ML can address challenges and advance biosensors in healthcare, bioresearch, environmental monitoring and medical diagnostics over conventional signal processing techniques.

3.Methodology

To achieve the objectives of the projects the following methodologies are followed sequentially.

3.1 Feature Extraction

3.1.1. Data collection and analysis of I-V transfer characteristics

The raw data collected is of an Organic Electrochemical Transistor(OECT) *Figure 3.1* based biosensor. Upon applying gate voltage to OECT, Ion's migration takes place from the electrolyte. Due to movement of ions, there will be change in conductivity. Due to applied voltage at gate, current starts flowing between the source and drain electrodes.

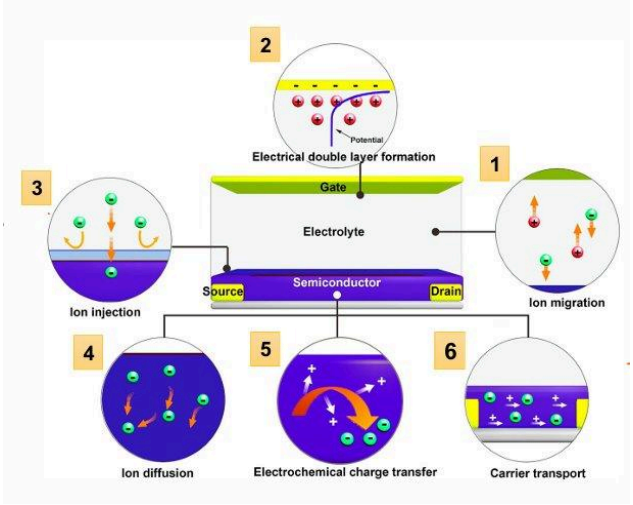


Figure 3.1 OECT working

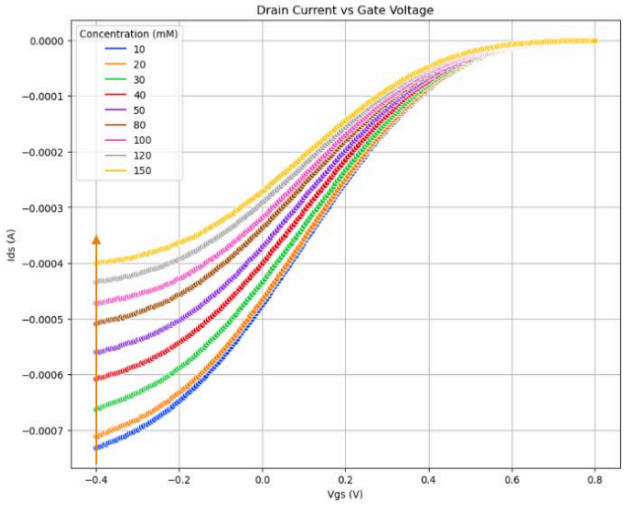


Figure 3.2 I-V transfer characteristics

Upon analysing the *I-V transfer characteristics* of the biosensor, *Figure 3.2*, it can be characterised as a *PMOS* device with drain to source and gate voltage of -0.5V and -0.4V respectively. The gate and drain to source voltage values are chosen such that the change in current is substantial and distinguishable with respect to change in concentration of ions. The Response is calculated with respect to concentration of the analyte, *Figure 3.3*.

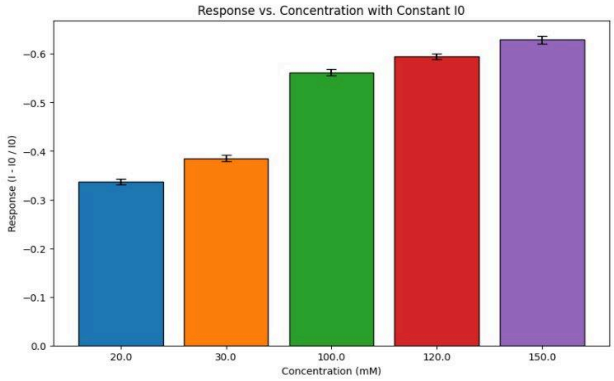


Figure 3.3 Conc. vs Response

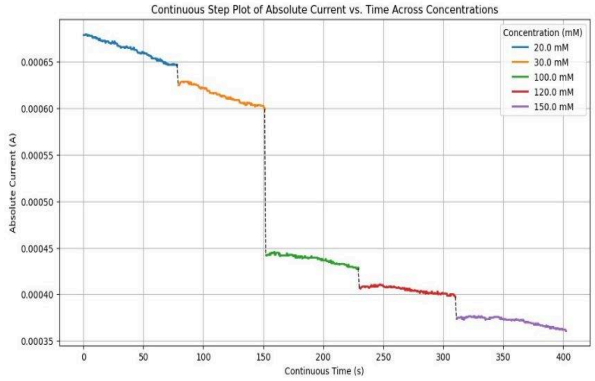


Figure 3.4 Conc. vs Drain current

3.1.2. Data preprocessing

The outliers, missing values and duplicate values are handled. Unnecessary columns, *gate voltage, gate current* and *drain voltage* are removed since the device is now operating at fixed voltage values. *Drain current(A), time(s)* and *concentration(mM)* are taken as features for further processing. Raw data is then *normalised* to maintain the consistency. Next we segmented the dataset into training, validation and testing sets for model training and validation.

3.1.3. Class imbalance and Noise Reduction

Synthetic Minority Over Sampling Technique(SMOTE), a technique in Machine Learning, specifically in classification problems that handles class imbalance by generating synthetic data points to increase the number of instances in the minority class and balancing all the classes of dataset, this helps in reducing the biases towards any classes in the dataset. New sample is created by choosing a random data point along the line joining the sample and one of its nearest neighbours. The below formula is used for generating the new sample.

$$new\_sample = sample_i + \lambda \times (sample_j - sample_i)$$

Wavelet transformation, a mathematical technique used to reduce noise in a signal and analyse data in frequency and time domains simultaneously. It is useful when signals have non-stationary features that change over time.

3.1.3. Dimensionality Reduction and Feature Extraction

The Principal component Analysis(PCA) is used to reduce the dimensionality of the data without losing the information in the dataset. PCA extracts principal components along which variance of data is maximum. It calculates the eigenvectors and projects the original data onto the new feature space designed by the selected principal components. It is generally used in Unsupervised Machine Learning.

3.2 Supervised Learning

Supervised learning algorithms are trained on a labelled dataset. The mapping of the inputs with output is captured by the algorithm and later utilised to make accurate predictions on new unseen data. The input features for the Unsupervised algorithm in this project are Drain Current(A) and

Time(s) and output feature is Concentration(mM). The model is initially trained on a training dataset and then validated and tested on validation and test datasets.

The matrices used for evaluating the supervised learning algorithms are:

1. *R-squared(R^2 - coefficient of determination)*: Indicates the goodness of the fit. Values range from 0 to 1, higher values indicated better model performance.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

2. *Mean Squared Error(MSE)*: The average of the squared differences between predicted and actual values. Smaller values indicate better model performance. It is more sensitive for outliers.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

3. *Root Mean Squared Error(RMSE)*: The square root of MSE. It is useful in understanding model performance on the original data scale.

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

4. *Mean Absolute Error(MAE)*: The average of the absolute difference between predicted and actual values. Smaller values indicate better model performance. It is a straightforward measure of accuracy.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

The Supervised algorithms used in this project are:

3.2.1. Linear Regression

This algorithm models a linear relationship, linear equation between input variables and single output variables. It is used when there is a linear relationship between input and output.

1. *Model Equation*:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \epsilon$$

where y: output variable, X: Input variable, β: Coefficient, ϵ: Error term.

2. *Cost function*: Minimises the the Residual Sum of Squares(RSS)

$$\text{RSS} = \sum(y_i - \hat{y}_i)^2$$

3. *Optimization*: Uses gradient descent or normal equations to find the best fitting parameters β.

3.2.2. Lasso Regression

It is Linear Regression with L1 regularisation to prevent overfitting. Used when there is high dimensional data.

1. *Model Equation*:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \epsilon$$

where y: output variable, X: Input variable, β: Coefficient, ϵ: Error term.

2. *Cost function*:

$$\text{Cost} = \text{RSS} + \lambda \sum_{i=1}^{n} |\beta_i|$$

where λ: regularisation parameter.

3. *Optimization*: Uses coordinate descent or subgradient to find the best fitting parameters β and L1 penalty.

3.2.3 Support Vector Regression(SVR)

It uses Support Vector Machine(SVM) concepts to predict continuous outcomes. Used to handle non linear relationship between input and output. It is robust to outliers and tolerates some amount of deviation. It constructs a hyperplane that maximises the margin while allowing some errors within the margin. The kernel function used for SVR in this project is Radial Basis function(RBF), a general purpose kernel for regression tasks, capable of handling non linear relationships in dataset.

1. *Model Equation*: calculate prediction using

$$f(x) = w^T \phi(x) + b$$

where $f(x)$: output variable, x: Input variable, w^T is weight vector, $\phi(x_i)$ is kernel function, $K(xi) = \phi(x_i)$.

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

2. *Optimization*: minimise

$$\frac{1}{2}||w||^2 + C\sum \xi_i$$

where ξ_i are slack variables.

3.2.4. Gaussian Process Regression

It is a probabilistic approach to regression. This algorithm treats the output variable as a distribution over possible functions. It is used when there is a complex non-linear relationship between input and output variables.

1. *Mean and Convergence function*: Prior

$$f \sim \mathcal{GP}(m(x), k(x, x'))$$

where y: output variable, X: Input variable, β: Coefficient, ϵ: Error term

2. *Posterior Distribution*: update the prior to obtain posterior using Bayes theorem.

$$p(f_*|X, y, X_*)$$

3.2.5 Decision Tree Regression(DTR)

It partitions the dataset into subsets based on feature values. It makes predictions by averaging the target values in the leaf nodes. DTR algorithms are used when there is a complex non-linear relationship between input and output variables and interpretability is important. Recursively splits data until converges to minimum samples per leaf.

1. Splitting criterion: minimise the variance within subsets.

$$\text{Variance} = \frac{1}{N}\sum(y_i - \bar{y})^2$$

3.2.6. Random Forest Regression(RFR)

It is an ensemble of DT that improves accuracy and reduces overfitting, by combining multiple DT to produce a robust final prediction.

1. *Bootstrap Sampling*: create multiple subsets of data and train a DT on each subset.
2. *Prediction*: Averages prediction from all DTs.

$$\hat{y} = \frac{1}{N}\sum_{j=1}^{N} T_j(x)$$

3.2.7. Gradient Boost Regression(GBR)

This algorithm sequentially trains Trees to correct and minimise errors of previous trees.

1. Loss Function: minimise MSE

2. Update Rule: Fit a tree and update it.

$$r_i = y_i - F_{m-1}(x_i)$$

$$F_m(x) = F_{m-1}(x) + \gamma T_m(x)$$

3.2.8. K-nearest neighbours(KNN)

KNN makes predictions based on the k closest training samples in the future space. It identifies the nearest data points and uses their outputs to predict the outputs of new data points. It is used when the dataset is relatively small and highly non-linear.

1. *Distance metric*: calculate euclidean distance

$$d(x, y) = \sqrt{\sum (x_i - y_i)^2}$$

2. *Prediction*: Fit a tree and update it.

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i$$

3.3 Unsupervised Learning

In contrast to supervised learning which involves predicting labels, unsupervised learning is a machine learning method where a model is trained on a dataset without target outputs. The primary goal of these Unsupervised Learning methods is to find the hidden or underlying patterns, groupings, or structures in the data especially when the labelled data is not available. Here Unsupervised Learning methods are applied on the dataset containing columns of Drain Current , Continuous time data and Concentration in mM. At the time of clustering, true labels (Concentration in mM) are not involved and only used in later stages for evaluating the performance of the algorithms. Since the identified clusters are used to compare with these true labels,we aim to assess how accurately the meaningful groups within the data are clustered by these algorithms. In this project we have used K-Means, Time Series K-Means, Hierarchical Clustering, DBSCAN, and Gaussian Mixture Models. Implementation of these unsupervised algorithms helps us to understand distinct behaviour and pattern, with each algorithm taking a distinct approach to measure the similarity and form groupings. Finally the performance of the algorithms will be compared with the help of visualisation and evaluation metrics such as Adjusted Rand Index (ARI) and Silhouette Score.

3.3.1 K-Means Clustering

K-Means Clustering is an Unsupervised Machine Learning algorithm that works by assigning data points to one of the nearest clusters depending on the distance between the datapoint and the cluster centre.This algorithm is majorly influenced by the no of the clusters(k) and the initial selection of centroids .But here we overcome this by setting the no of clusters to the count of unique concentrations in the dataset. Also by using the kmeans++ initialization method which helps us in initialising the placement of centroids away from each other, majorly helping us to reduce the sub-optimal solutions and improving computational efficiency.

Working of K-Means Clustering:-

1. *Initialization* : Randomly initialise k cluster centroids
2. *Assignment step*: Assign each data point to the nearest centroid based on the euclidean distance.
3. *Update step*: Involves recalculation of centroid based upon the mean of all the data points assigned to the respective cluster.
4. *Convergence*: Repeat the above Assignment and Update step until the centroids no longer change significantly or maximum number of iterations are reached.

K-Means is computationally efficient algorithm and converges relatively fast, when compared complex clustering methods.As discussed earlier the K-means is sensitive to the initialization of centroids leading to different outcomes on placement of initial centroids differently.Also this algorithm performs well when are clusters spherical and evenly distributed.The major disadvantage is ,it performs poorly in the presence of outliers,indicating to us this algorithm is sensitive to the outliers.

3.3.2 Hierarchical Clustering

Hierarchical Clustering is an unsupervised learning method used to build a hierarchy of clusters without the need of specifying the the number of clusters initially,but as we already know the required no of clusters which is equal to the count of unique concentration in the dataset,we will fix this parameter.It generally creates a tree-like structure called dendrogram, that helps in visualising the merging or splitting of clusters at different levels.We will be following a Agglomerative clustering which is a bottom-up approach,it starts by considering each datapoint as single cluster, and it merges into pair of clusters based on the similarity of clusters, until we reach the desired no of clusters.

Working of Hierarchical Clustering:-

1. *Initialization* : Each datapoint is considered as single cluster
2. *Similarity Calculation*: Similarity or distance calculation takes place,based on the linkage criteria. There are several linkage criteria to determine the distance between the clusters,like
    - "single - Distance between the closest point of two clusters"
    - "maximum or complete- Distance between the farthest points of two clusters",
    - "average - Average distance between all points in two clusters",
    - "ward - Minimises the variance within clusters during merging".
3. *Merging of Clusters*: Merging takes place based on the linkage criteria the closest pair gets merged into a single cluster.
4. *Iteration* : We iterate step 2 and 3,until we reach our required number of clusters.
   This algorithm is not computationally effective,making it not suitable for large datasets.

3.3.3 DBSCAN Clustering

DBSCAN (Density-Based Spatial clustering of Applications with Noise) is an Unsupervised algorithm that groups data based on the density.In contrast to the K-Means and Hierarchical it doesn't require specifying the number of clusters.It mainly depends on the two parameters epsilon and min_samples, to define dense regions and determine the clustering structure.

Working of DBSCAN Clustering:-

1. Based upon the values of epsilon and min_samples, a point is further classified into a Corepoint, Borderpoint, and Noise.As these parameters help to identify the neighbours.
    - *Epsilon* : The radius,within which data points are considered as neighbours.
    - *Min_samples* : The minimum number points within the radius of epsilon value to form a dense region
    - *Corepoint*: A point is considered to be a corepoint, if it has a data points count equal to min_samples with the radius of value equal to epsilon.
    - *Borderpoint*: A point with in the epsilon but doesn't have enough no of min_samples to consider it has a corepoint
    - *Noise*: Points that are neither core points nor border points can be classified as noise.
2. Clusters are created by connecting core points that are within epsilon distance of each other, along with any border point that are directly reachable from these corepoints.

This clustering has an ability to detect clusters of arbitrary shapes making it suitable for complex datasets,and potential to effectively identify the noise.It is most sensitive to the min_samples and the epsilon value,appropriate tuning of these parameters is crucial for optimal clustering results.

3.3.4 Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a probabilistic model that assumes the data is generated from a mixture of several gaussian distributions.Compared k-Means where data points are assigned on basis of distance ,here the data points are assigned to a cluster with a probability.

Working of Gaussian Mixture Model:

1. *Initialize*: Need to specify the required no of clusters, and other parameters like type of covariance
2. *Expectation-Maximization (EM) Algorithm*: The gaussian model uses this algorithm iteratively to adjust model parameters to fit the data.
    - *Expectation*: This step involves calculation of probability of each datapoint that it belongs to a gaussian component, involving the use of Probability density function.
    - *Maximizations*: This step involves updating of parameters (mean,covariance,mixing coefficients)for each cluster based on the probabilities calculated in the Expectation step.Updation of these parameters take place to maximise the likelihood.
3. *Convergence*: Repeat the step 2 that is Expectation-Maximization step until the parameters converge.
4. *Clustering*: Each data point is assigned to the gaussian component with the highest probability.

This model has the ability to clusters of varying shapes and densities. May not perform well if the data doesn't follow a Gaussian distribution and is sensitive to the parameter intilization.

3.3.5 Time Series K-Means Clustering

Time Series K-means Clustering is an adaptation of K-Means for clustering time-series data.This algorithm works by grouping time-series data based on the similarity of their patterns.This algorithm is specifically designed for temporal data,and has ability to handle variable time-series length and patterns.

Working of Time Series K-Means Clustering:

1. *Initialization* : set the no of clusters and distance metrics such as dtw or soft-dtw.

2. *Cluster Assignment* : Computation of distance using the specified metric from each time series to each cluster takes place and assigns it to the cluster with the closest centroid.
3. *Centroid Update*: Update the cluster centroids by calculating the average of time-series while maintaining their temporal structure.This means calculating a representative time series that minimises the distance to all series assigned to that cluster, preserving the sequence alignment.
4. *Convergence* : Repeat the step 2 and step 3 until the centroids stop changing by a significant amount.

This algorithm is computationally expensive ,so it requires careful selection of distance metrics. The distance metric like dtw (distance time wrapping) measures the similarity between two sequences by stretching or "warping" them in time to find the minimum cumulative distance between them. This method is helpful for time series data where patterns are similar but appear at slightly different time points.Soft-Dynamic Time Warping (Soft-DTW) is an advanced variant of the traditional Dynamic Time Warping (DTW) algorithm.

3.4 LSTM

LSTM (Long Short Term Memory)'s are a class of recurrent neural networks specifically designed to handle sequential data with temporal dependencies,which makes them more effective for the mission involving time-series prediction or classification.Here we are we will preprocess the dataset into scaled sequences and utilising an LSTM architecture to classify concentration levels based on the drain current and Continuous Time data.The benefit of using LSTM is ,it has ability to capture long-term dependencies in sequential data as they incorporate memory cells and gates to retain important information over time, allowing them to learn complex temporal patterns.

The data will be transformed into sequences of time-series data, each of these sequences are of fixed length,meaning fixed no of readings.Using these sequences LSTM learns about the temporal dependencies involved and outputs a prediction for the concentration level.

Working of Long Short Term Memory:

1. *Preprocessing* : In this we are converting the dataset into feasible data that can be fed into the LSTM.

- One-hot Encoding: Performed one-hot encoding to convert the target labels that is concentration (mM) into binary format.It basically assigns 0's and 1's to each unique category present in the target variable.
- Normalisation : The dataset is then scaled using MinMaxScaler().
- Sequences: Sequences of length =10 are prepared for the LSTM input.

2. *Building of Model*: In this step we will be creating an LSTM based neural network that consists of LSTM layers,dropout(to prevent overfitting) and dense layers for output ,the epoch size is set to 50 and batch size is 32.
3. *Training of Model*: We will be splitting the dataset into train and test data.Then will be using the built model on the train dataset with the help of Adam optimizer and categorical cross-entropy loss function.Also we will be involving the metrics like accuracy and loss over epochs.

4.Results and Discussions

4.1 Supervised Learning

The supervised algorithms are trained and tested on two different datasets of the same device. Training set of Clean dataset *Figure 4.1*, a perfect dataset with very minute noise is used to train and validate. Then models are tested on noisy data *Figure 4.2* to validate the robustness and reliability of ML algorithms. The performances of supervised machine learning algorithms on clean data are as in *Table 4.1* and on noise data are as in *Table 4.2*.

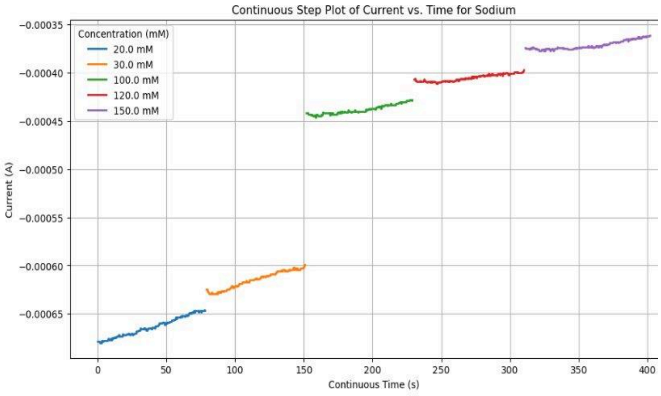


Figure 4.1 : Clean data of device

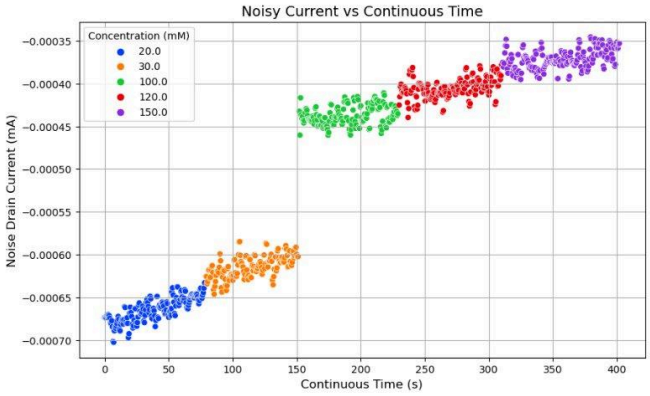


Figure 4.2 : Noisy data of device

Model Name	R^2 Score	MSE	RMSE	MAE
Linear Regressor	0.971	76.49	8.74	8.08
Lasso Regression	0.970	77.23	8.78	8.05
Support Vector Regressor	0.987	32.59	5.70	4.51
Gaussian Process Regressor	0.989	27.18	5.21	2.77
Gradient Boosting Regressor	0.999	1.87	0.001	0.001
Decision Tree Regressor	1.00	0.00	0.00	0.00
Random forest Regressor	1.00	0.00	0.00	0.00
K-nearest neighbours	1.00	0.00	0.00	0.00

Table 4.1 : Results of Supervised algorithms on clean data

It is observed that from *Figure 4.2, Figure 4.4, Table 4.1 and Table 4.2* among all the proposed supervised models *DTR, GBR, RFR and KNN* models are performing very well on both Clean and Noisy datasets with R^2 *score* > 0.99. There are very low errors for these algorithms. The results show that the models are adaptive to noisy data and able to handle non-linear data. However, *Linear regression, Lasso regression, SVR, GPR* are having high R^2 *score* > 0.96 but the error rate is higher and further increasing for noisy data.

Model Name	R2 Score	MSE	RMSE	MAE
Linear Regressor	0.967	85.90	9.26	7.92
Lasso Regression	0.966	86.77	9.31	7.83
Support Vector Regressor	0.982	45.40	6.73	4.70
Gaussian Process Regressor	0.941	153.88	12.40	3.42
Gradient Boosting Regressor	0.997	6.241	2.49	0.22
Decision Tree Regressor	0.9976	6.242	2.49	0.22
Random forest Regressor	0.9975	6.38	2.52	0.24
K-nearest neighbours	0.9973	6.96	2.63	0.26

Table 4.2 : Results of Supervised algorithms on noise data

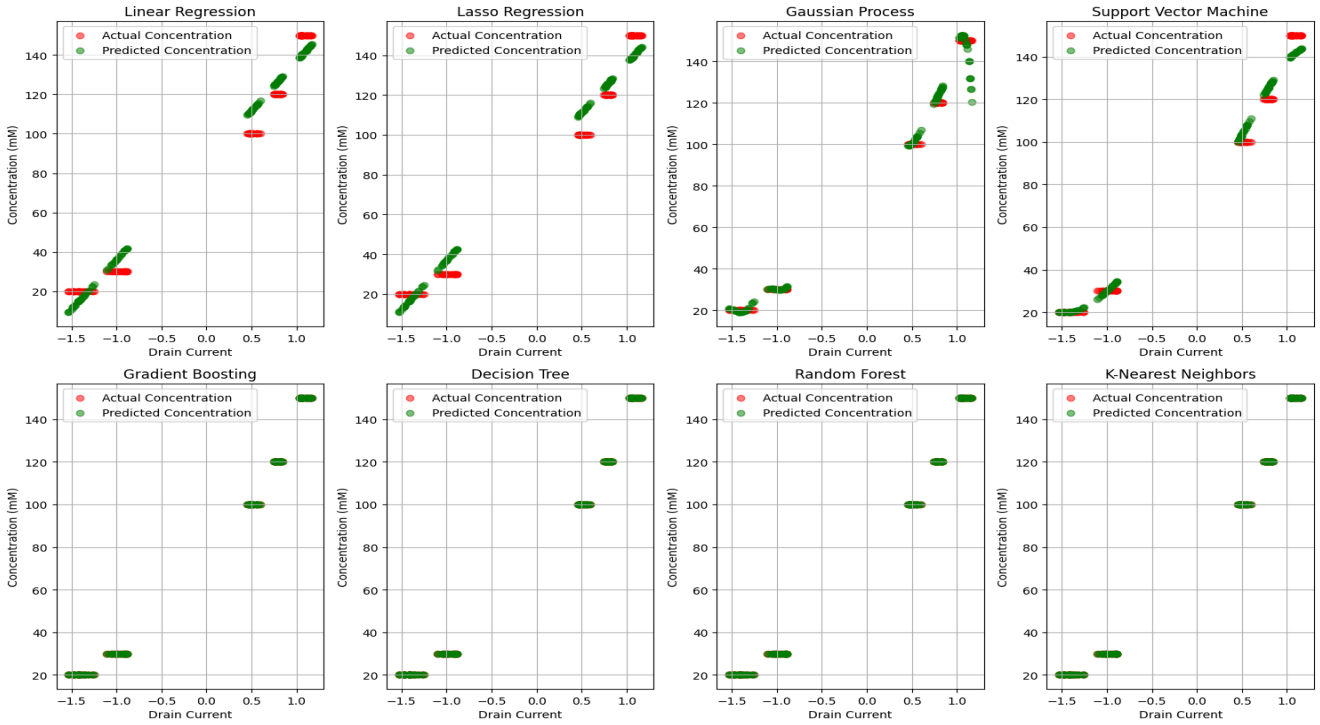


Figure 4.1 : Actual vs Predicted Conc. by Supervised algorithms on clean data

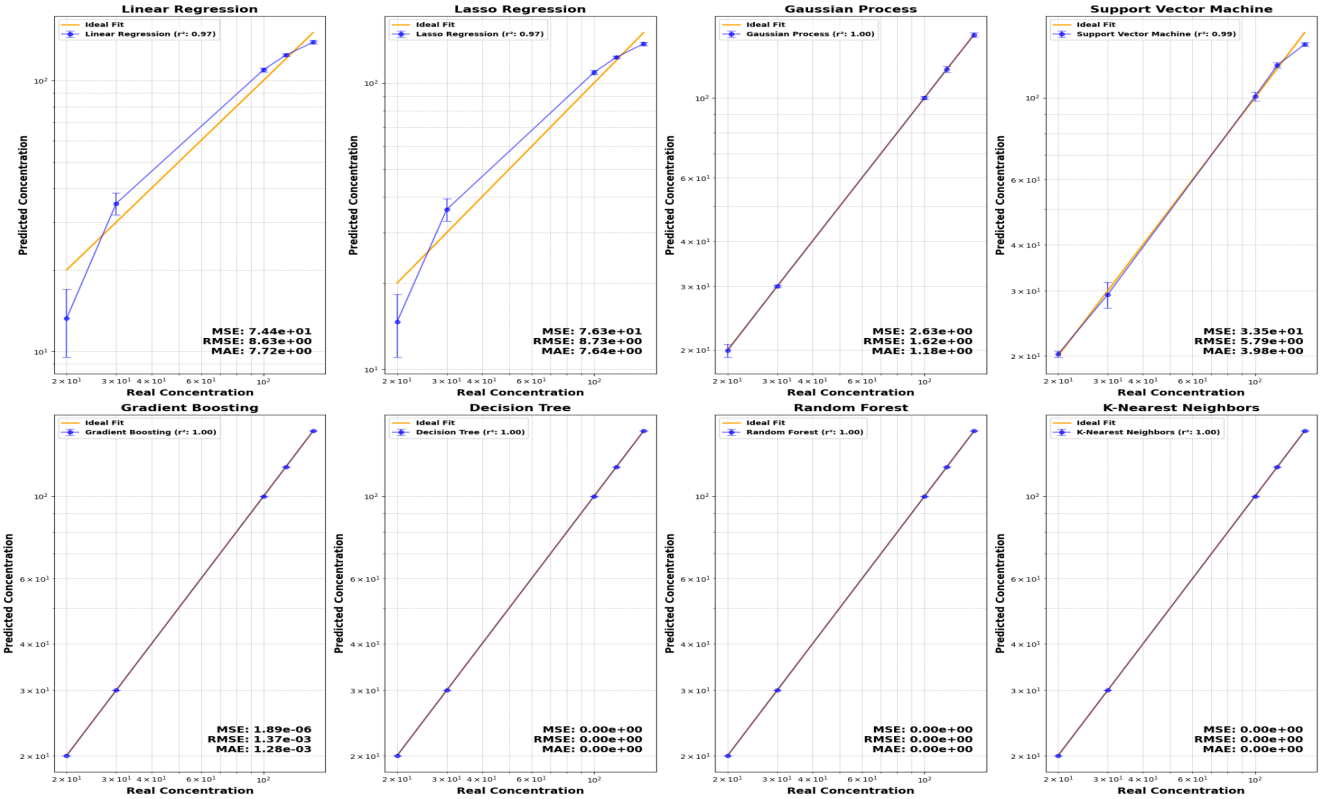


Figure 4.2 : Calibration curves of Supervised algorithms on clean data

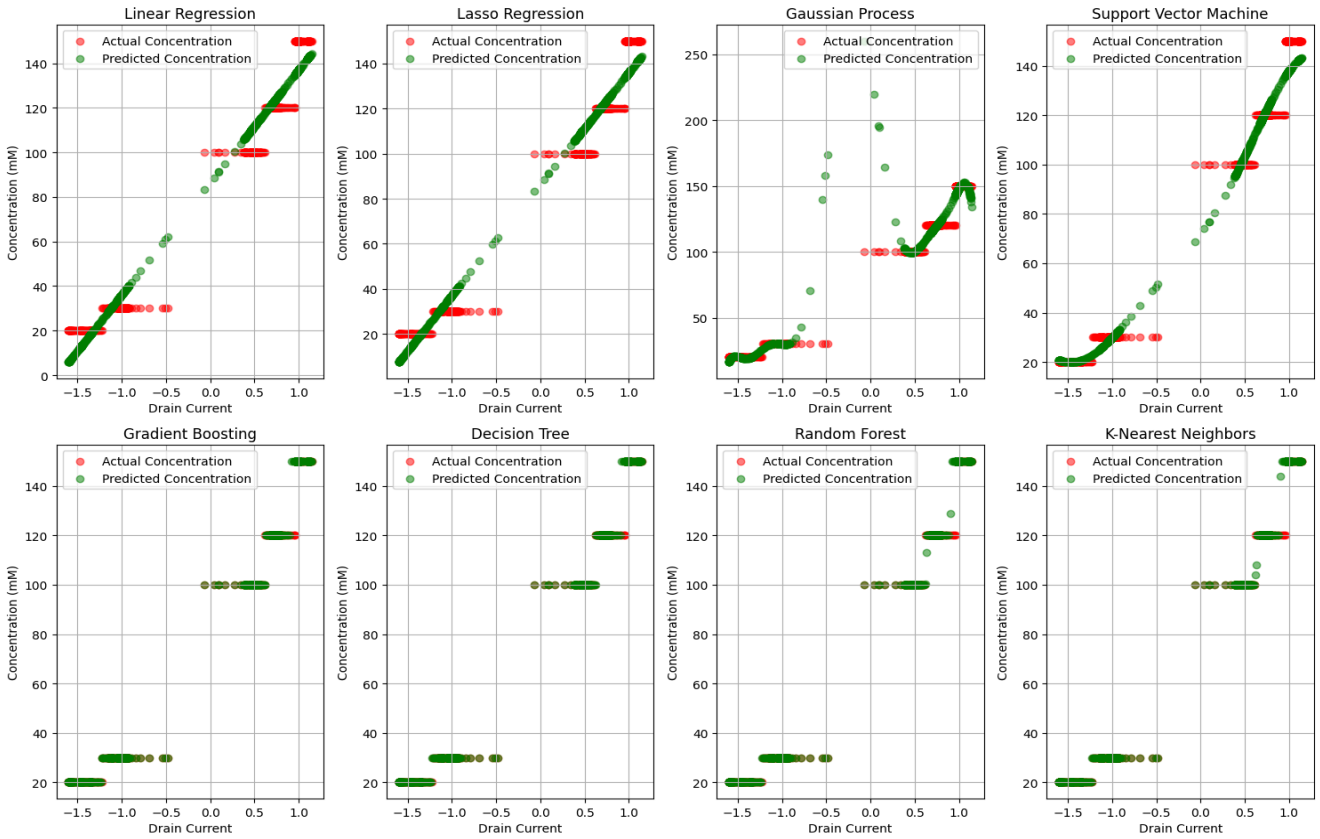


Figure 4.3 : Actual vs Predicted Conc. by Supervised algorithms on noisy data

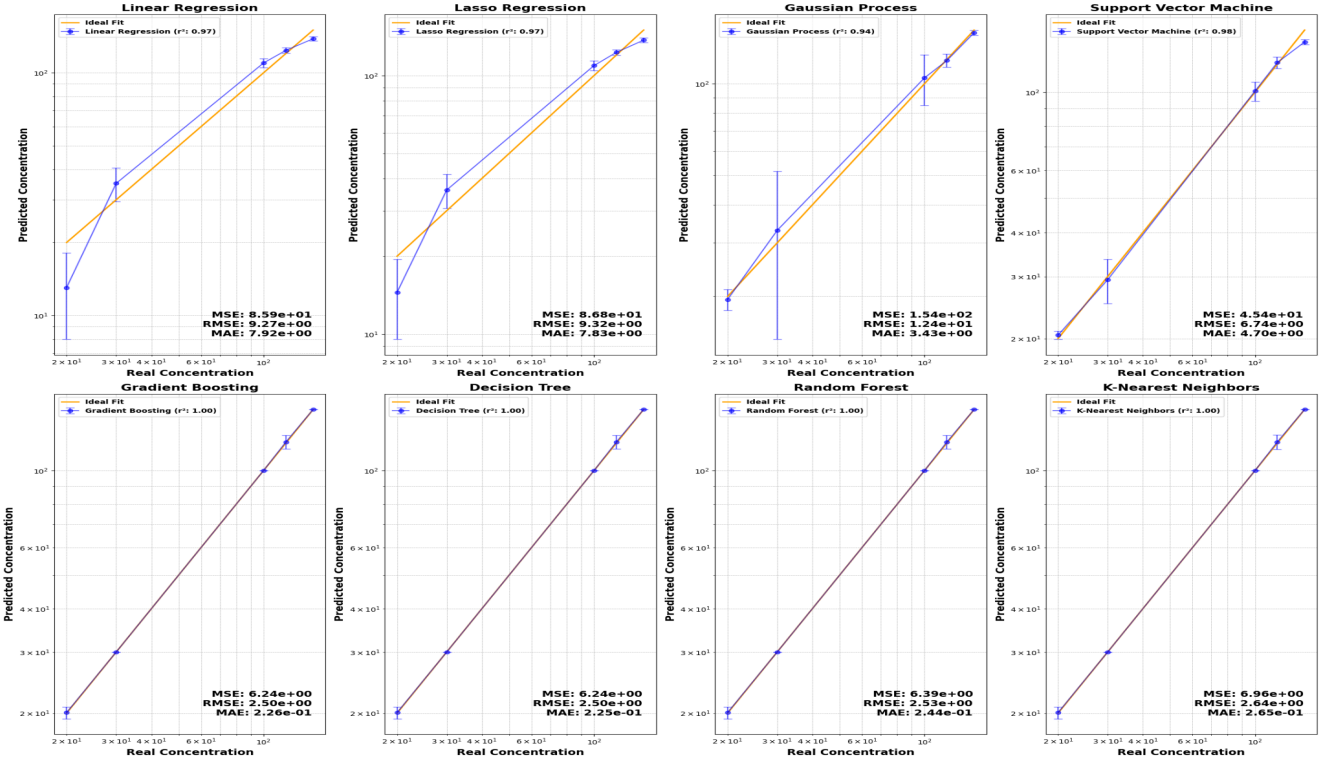


Figure 4.4 : Calibration curves of Supervised algorithms on noise data

4.2 Unsupervised Learning

Model name	ARI (Clean Data)	ARI (Noisy Data)	ARI (K+ Data)	Silhouette Score (Clean Data)	Silhouette Score (Noisy Data)	Silhouette Score (K+ Data)
K-Means	0.8347	0.8161	0.5518	0.5750	0.560	0.5418
DBSCAN	1	1	0.5885	0.6220	0.560	0.3601
Gaussian Mixture Model	0.7878	0.7877	0.5716	0.5577	0.559	0.5294
Time-series K-Means -dtw	0.9374	0.8642	0.5716	0.6074	0.5605	0.5427
Time-series K-Means soft-dtw	0.7873	0.8161	0.5885	0.5591	0.5603	0.5434
Hierarchical	1	1	1	0.6220	0.5607	0.5294

Table 4.3: Results of Unsupervised algorithms on different data sets

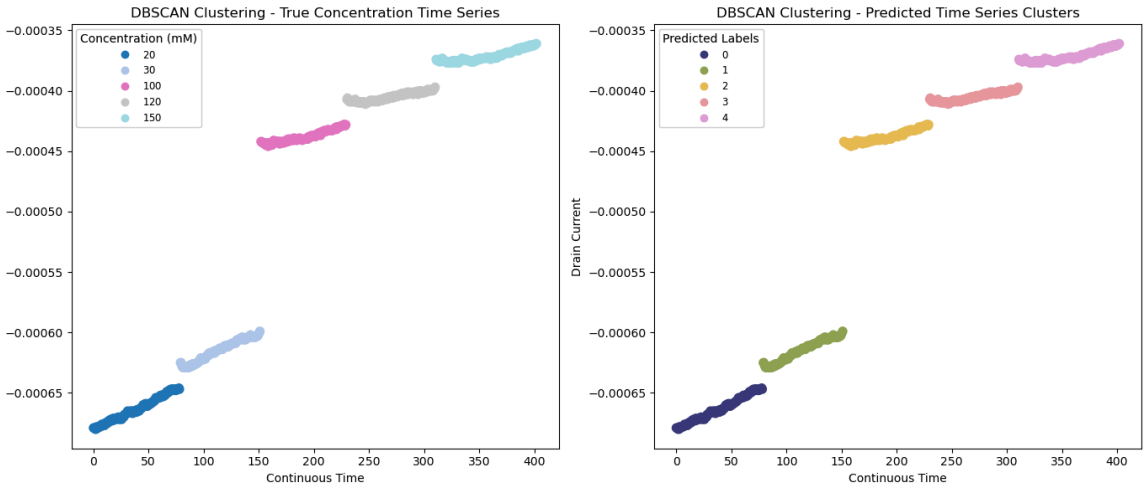


Figure 4.5 Comparison of True and Predicted Clusters of DBSCAN on Clean Data

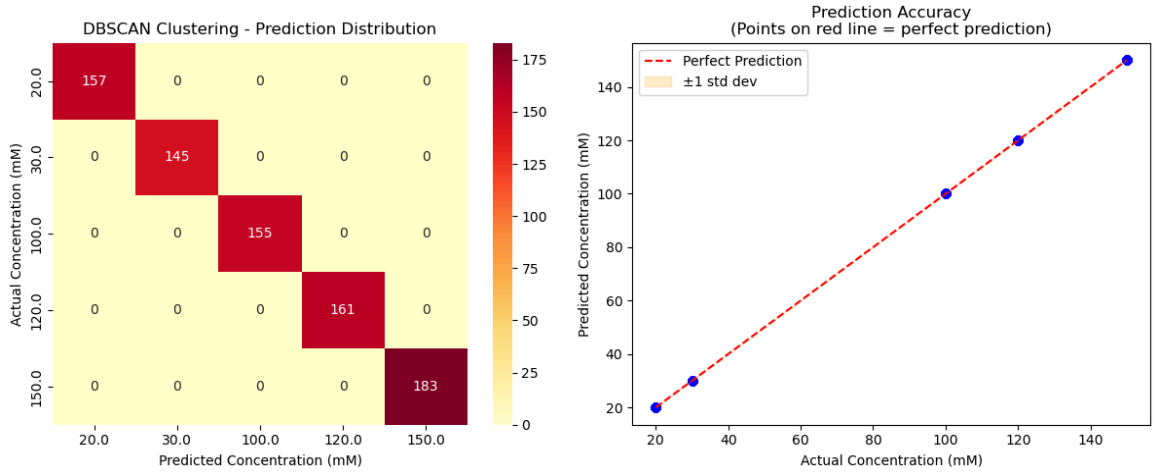


Figure 4.6 Confusion Matrix and Prediction Accuracy plot for DBSCAN on Clean data

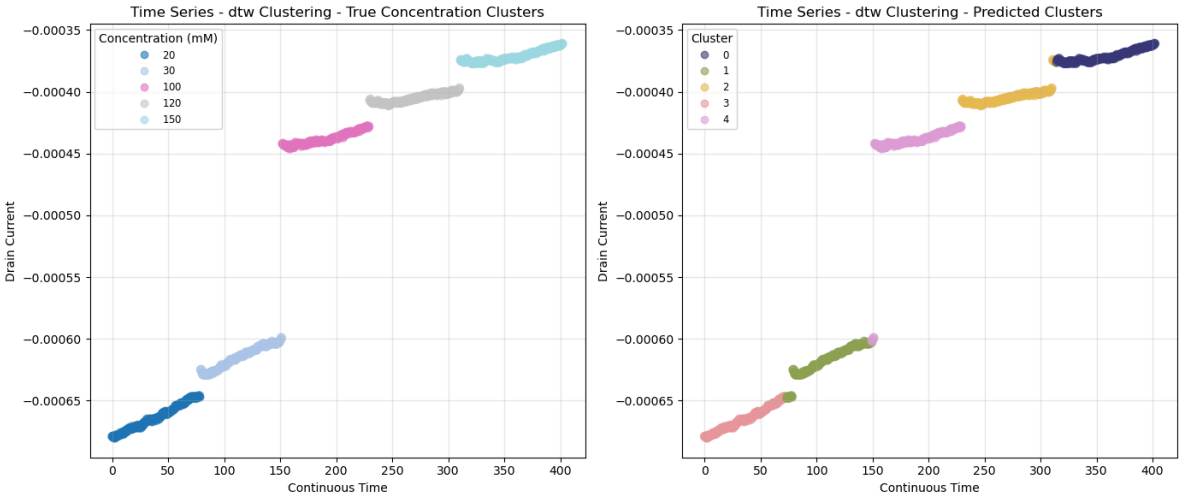


Figure 4.7 Comparison of True and Predicted Clusters of TimeSeries-dtw on Clean Data

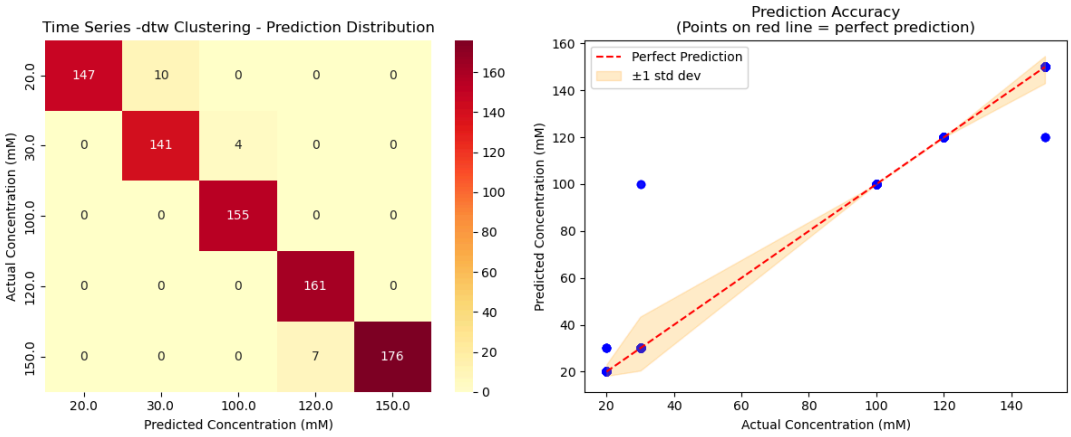
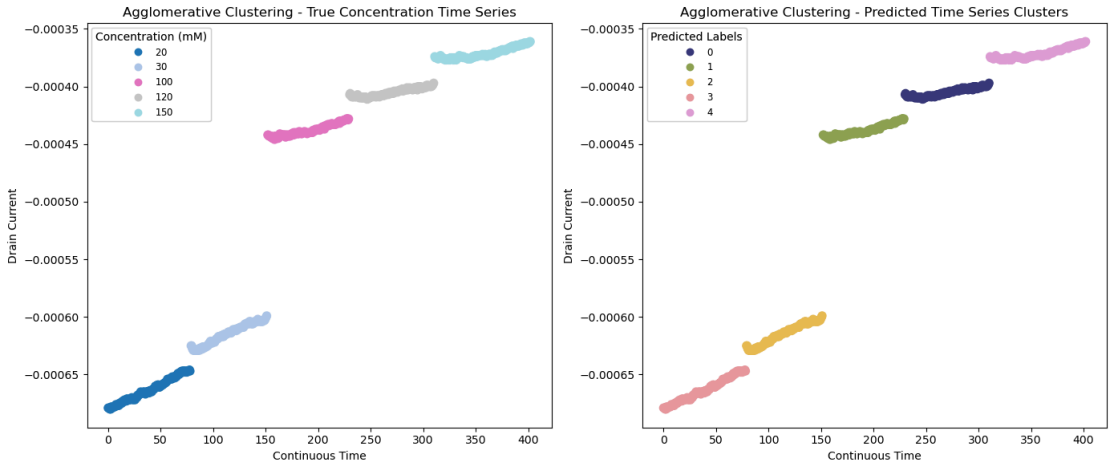


Figure 4.8 Confusion Matrix and Prediction Accuracy plot for TimeSeries-dtw on Clean data



.Figure 4.9 Comparison of True and Predicted Clusters of Hierarchical on Clean Data

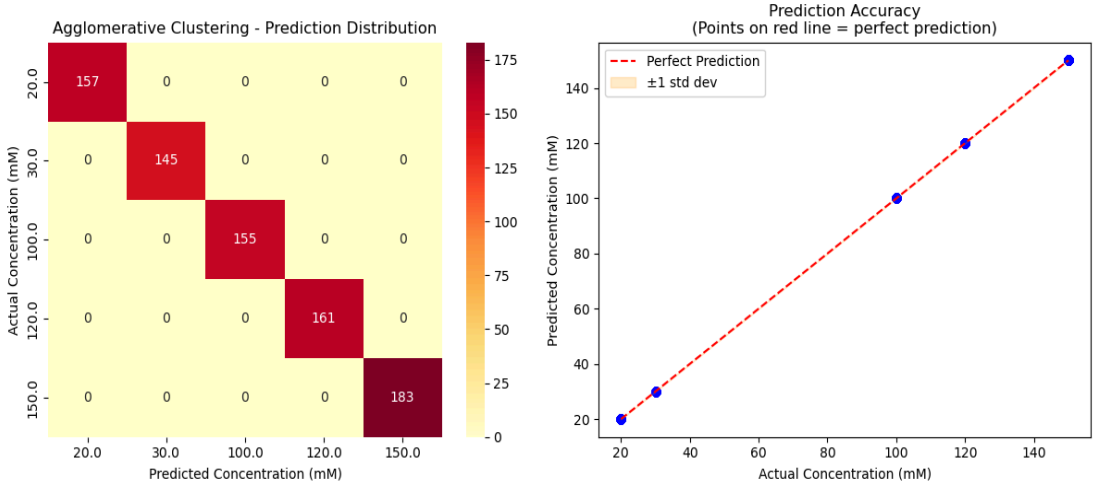
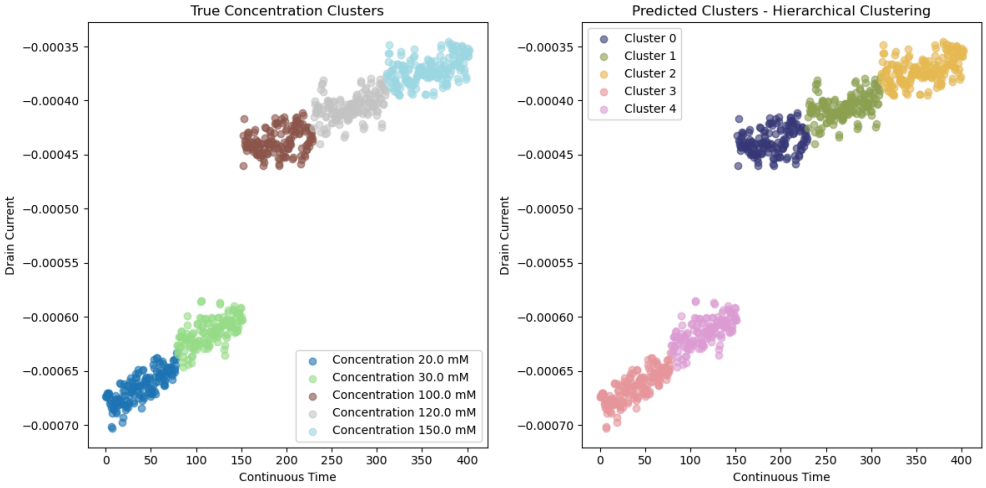
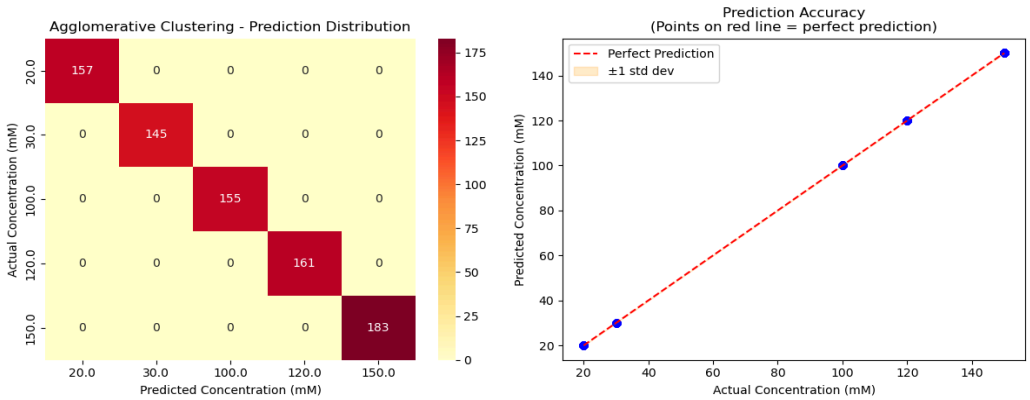


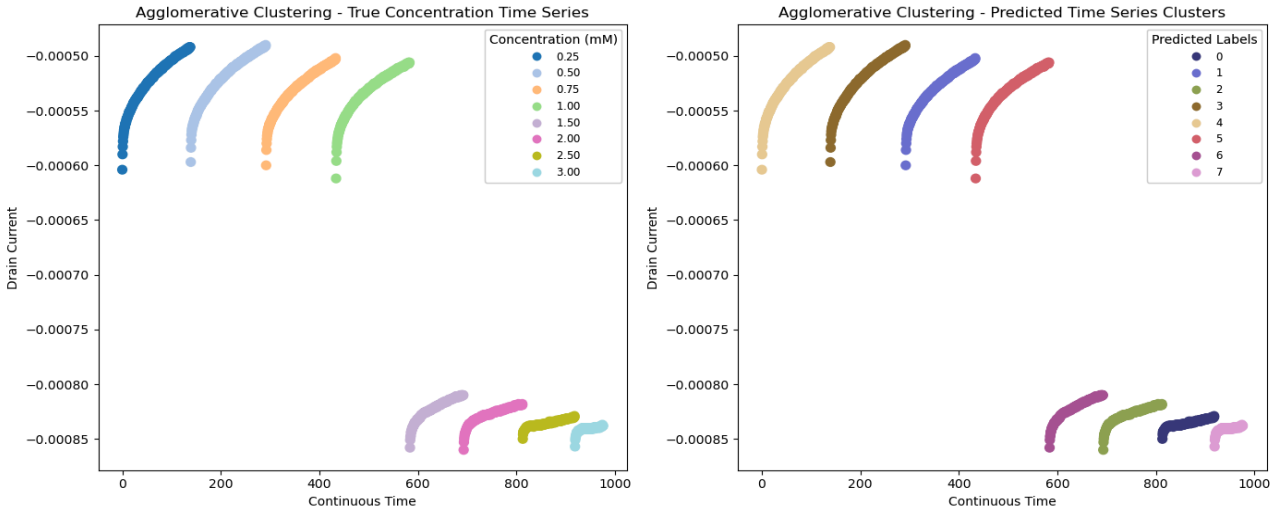
Figure 4.10 Confusion Matrix and Prediction Accuracy plot for Hierarchical on Clean data



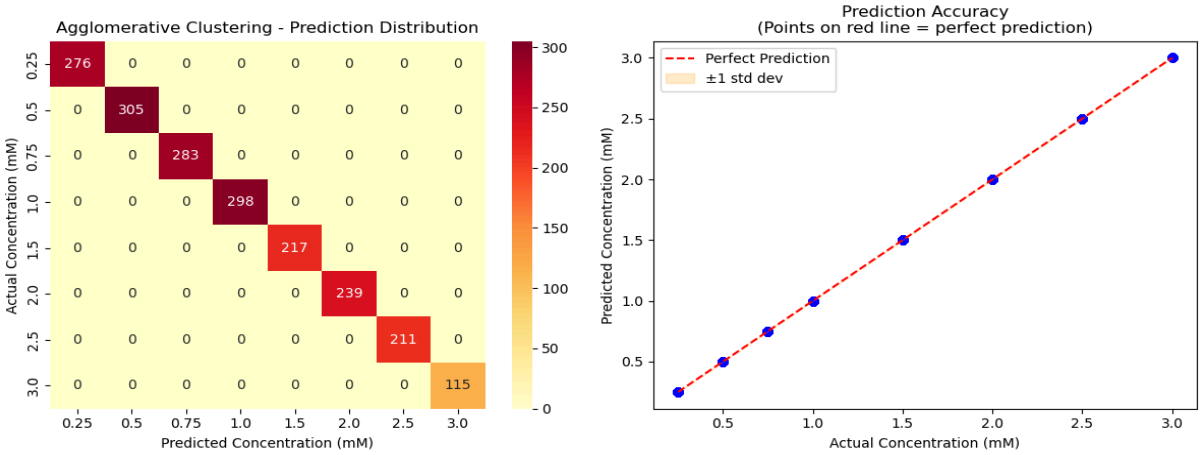
4.11 Comparison of True and Predicted Clusters of Hierarchical on Noisy Data.



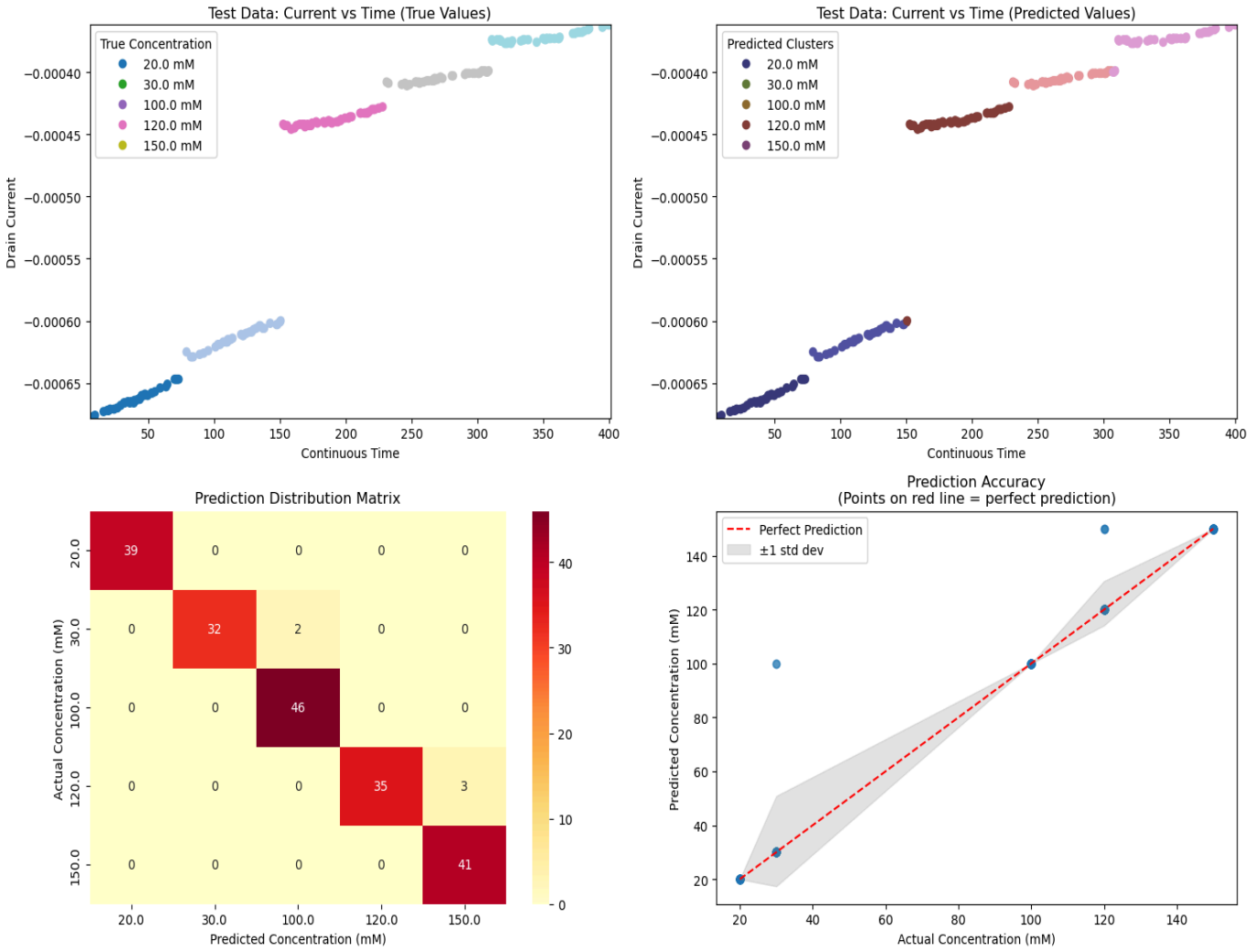
4.12 Confusion Matrix and Prediction Accuracy plot for Hierarchical on Noisy data



4.13 Comparison of True and Predicted Clusters of Hierarchical on K+ Data.



4.14 Confusion Matrix and Prediction Accuracy plot for Hierarchical on K+ data



4.15 Comparison of True and Predicted Clusters of LSTM on Clean Data and Confusion Matrix and Prediction Accuracy plot for LSTM on Clean data

Upon observation of *Table 4.3*, the analysis reveals varying performance patterns across different clustering methods. Both traditional (*K-Means, GMM*) and time-series variants (*DTW K-Means*) demonstrate relatively good performance on clean data, with ARIs ranging from 0.787 to 0.9374. The *DTW K-Means* particularly excels with clean data (ARI: 0.9374) and shows improved noise handling capabilities (ARI: 0.8642 on noisy data) compared to traditional methods. However, a significant limitation emerges when these models are applied to potassium-enriched data, where performance deteriorates substantially (ARI: ~0.55-0.57). This consistent degradation across both traditional and time-series variants suggests fundamental limitations in their ability to adapt to complex

dataset conditions of potassium(K+), despite their computational efficiency and good performance with clean data.

Advanced clustering solutions show remarkable improvements. *DBSCAN* and *Hierarchical Clustering* achieve perfect ARI scores (1.0) on clean and noisy data, with *Hierarchical Clustering* maintaining this performance even on k+ data. This demonstrates superior stability and adaptability compared to earlier methods.While *DBSCAN* exhibits excellent noise-handling capabilities and achieves the highest silhouette score (0.6220) on clean data, its performance declines significantly on k+ data with a drop in silhouette score to 0.3601, indicating challenges in maintaining cluster density in complex potassium dataset as it contains nonlinearly related input output features.. In contrast, *Hierarchical Clustering* maintains perfect ARI scores (1.0) across all datasets, including k+ data, while maintaining consistent silhouette scores. This remarkable stability and adaptability of Hierarchical Clustering in scenarios requiring robust clustering performance over very complex and nonlinearly related input output features. The *LSTM* model represents the peak of performance, achieving consistently high accuracy across all conditions (97.47% clean, 96.97% noisy, 97.73% K+data). Its minimal performance variation (<0.5%) and robust handling of temporal patterns make it the superior choice for applications requiring high precision and reliability, while traditional clustering methods remain valuable for scenarios prioritising interpretability and computational efficiency.

6.Conclusion

It is validated that the proposed ML based approaches in this project have the ability to enhance, transform and even replace conventional signal processing techniques, making it more reliable and efficient. The challenges faced by traditional techniques are addressed and solved by ML algorithms. Computations like Fourier transforms, Analog to Digital (ADC) conversions, Filters(HPF, LPF), Digital signal processing, fixed rule based logics are completely removed. The signal processing stages such as noise filtering, feature extraction and decision making are completely taken care of by ML algorithms. The results obtained demonstrated the potential performance of these algorithms. The biosensor systems can now handle noisy, complex raw data more efficiently and maintain high performance even in dynamic environments. The presented approaches can be implemented in real time, wearable and implementable biosensors like ECG patches, Continuous glucose monitors, point of care testing, remote health monitoring, precision diagnostics and bioresearch.

7.References

[1] Cui, Feiyun, Yun Yue, Yi Zhang, Ziming Zhang, and H. Susan Zhou. "Advancing biosensors with machine learning." ACS sensors 5, no. 11 (2020): 3346-3364.

[2] Schackart III, Kenneth E., and Jeong-Yeol Yoon. "Machine learning enhances the performance of bioreceptor-free biosensors." Sensors 21, no. 16 (2021): 5519.

[3] Wu, Wanqing, Jianlei Yang, Yu Zhou, Qinggong Zheng, Qing Chen, Zhaoao Bai, and Jiaqi Niu. "Chemometrics‐based signal processing methods for biosensors in health and environment: A review." Electroanalysis 36, no. 7 (2024): e202300207.

[4] Singh, Anoop, Asha Sharma, Aamir Ahmed, Ashok K. Sundramoorthy, Hidemitsu Furukawa, Sandeep Arya, and Ajit Khosla. "Recent advances in electrochemical biosensors: Applications, challenges, and future scope." Biosensors 11, no. 9 (2021): 336.

[5] Zhang, Kaiyi, Jianwu Wang, Tianyi Liu, Yifei Luo, Xian Jun Loh, and Xiaodong Chen. "Machine learning‐reinforced non invasive biosensors for healthcare." Advanced Healthcare Materials 10, no. 17 (2021): 2100734.

[5] Aghabozorgi, S., Seyed Shirkhorshidi, A., Ying Wah, T., 2015. Time-series clustering - A decade review. Inf. Syst. 53, 16–38. doi: 10.1016/j.is.2015.04.007.

[6] W. Zhang et al., "LSTM-Based Analysis of Industrial IoT Equipment," in IEEE Access, vol. 6, pp. 23551-23560, 2018, doi: 10.1109/ACCESS.2018.2825538.keywords: {Time series analysis;Sensors;Predictive models;Data models;Monitoring;Internet of Things;Power generation;Time series prediction;LSTM model;power equipment;industry Internet of Things},

[7] Sharma, Sumit, Debashree Kar, Akshay Moudgil, Samaresh Das, and Prashant Mishra. "Tungsten oxide thin film field-effect transistor based real-time sensing system for non-invasive oral cancer biomarker detection." Sensors and Actuators B: Chemical 407 (2024): 135486.

[8] Li, Ting, Jie Yan Cheryl Koh, Akshay Moudgil, Huan Cao, Xihu Wu, Shuai Chen, Kunqi Hou et al. "Biocompatible ionic liquids in high-performing organic electrochemical transistors for ion detection and electrophysiological monitoring." *ACS nano* 16, no. 8 (2022): 12049-12060.

8.Contributions :

Megavath Pavan : Worked on the background study of project, Exploratory data analysis and Supervised Learning.

Pujari Bheemesh: Worked on Unsupervised Learning and LSTM.