scientific reports

Check for updates

OPEN

Evaluating algorithms of decision tree, support vector machine and regression for anode side catalyst data in proton exchange membrane water electrolysis

Mahdi Arjmandi[1], Moslem Fattahi[1,2]✉, Mohsen Motevassel[1] & Hosna Rezaveisi[3]

Nowadays, due to the various type of problems stemmed from using chemical compounds and fossil fuels which have widely influence on whole environment including acid rain, polar ice melting and etc., number of researches have been leading on replacing the nonrenewable energy sources with renewable ones in order to produce clean fuels. Among these, hydrogen emerges as a quintessential clean fuel, garnering substantial attention for its potential to be synthesized from the electric power generated by renewable sources like nuclear and solar energies. This is achieved through the employment of a proton exchange membrane water electrolysis (PEMWE) system, widely recognized as one of the most proficient and economically viable technologies for effecting the separation of H_2O into H^+ and OH^-. In this study, the important affecting parameters on the anode side of catalyst in PEMWE and analyzed them by machine-learning (ML) algorithms through developing a data science (DS) procedure were discussed. Various machine learning models were subjected to comparison, wherein the Decision Tree models, specifically those configured with maximum depths of 3 and 4, emerged as the optimal choices, attaining a perfect 100% accuracy across both Dataset 1 and Dataset 2. Moreover, notable enhancements in accuracy values were observed for the Support Vector Machine (SVM) model, registering increments from 0.79 to 0.82 for Dataset 1 and 2, respectively. In stark contrast, the remaining models experienced a decrement in their accuracy scores. This phenomenon underscores the pivotal role played by the data generation process in rendering the models more faithful to real-world scenarios.

In recent years, concerns about global warming and its various environmental impacts, such as polar ice melting, acid rain, and rising sea levels, have become a primary focus for scientists. These issues are largely attributed to the consumption of refractory chemical compounds, particularly fossil fuels like coal, oil, and natural gas, as well as concerns about their depletion[1]. Consequently, there has been a concerted endeavor to expedite the advancement of renewable energy generation, storage, and conversion infrastructures, in light of projections indicating a prospective global power demand of approximately 30 and 46 TW by the years 2050 and 2100, respectively[2]. However, a major challenge in using solar and wind energy as renewable sources is their unscheduled and intermittent supply, which often does not match the grid power demands[3]. To address this issue, efficient systems for storing excess electricity must be developed. One promising approach is the use of electrocatalytic systems, which convert electricity into chemical energy for indirect storage of excess renewable energy[4]. Amongst a plethora of electrocatalytic technologies, water electrolysis stands out as the most efficacious means for producing pristine green hydrogen, harnessing the potential of renewable energy sources like solar and wind energies[5]. Pertinently, Pourrahmani et al. have undertaken a thorough inquiry into the feasibility and efficacy of employing PEMFC as an indirect means of water electrolysis storage, a process entailing the conversion of excess electricity generated by wind turbines into hydrogen gas[6]. This green hydrogen can be stored and used in the chemical industry, or for electricity production via fuel cells or internal combustion engines, with zero post-combustion pollutants[7]. Commercially available systems for water electrolysis include alkaline, solid

[1]Chemical Engineering Department, Abadan Faculty of Petroleum Engineering, Petroleum University of Technology, Abadan, Iran. [2]Department of Chemical and Materials Engineering, University of Alberta, Edmonton, AB, Canada. [3]Chemical Engineering Department, Faculty of Engineering, Razi University, Kermanshah, Iran. ✉email: moslem.fattahi@gmail.com

oxide and proton exchange membrane (PEM) electrolyzers, with the latter being more advantageous due to their more compact design, absence of leaking issues, higher current density, higher operating temperature and characteristic of high temperature potential resulting in higher energy conversion efficiency, greater hydrogen generation rate, lower gas crossover rate resulting in decreasing power consumption, part-load operating ability, and ability to operate at higher pressures due to their strong cell structure[8–11]. In contemporary times, there exists a burgeoning inclination towards the refinement and reconfiguration of alkaline water electrolyzers, coupled with the advancement of proton exchange membranes with applicability spanning both water electrolysis units and fuel cells. These strides stem from notable advancements witnessed within the domain of high-temperature solid oxide technology, as substantiated by research studies[12–14].

Water electrolysis is a process that water molecules spilt in hydrogen and oxygen gases using electricity through electrochemical process resulting in producing clean energy with no emission of pollution. The basic equation of water electrolysis is as Eq. 1[15].

$$H_2O(liquid) + Energy \rightarrow H_2(g) + {}^1/_2O_2(g) \tag{1}$$

The simplest water electrolysis system has been displays in Fig. 1a which consisting of an anode and a cathode connected through an external power supply and immersed in a conducting electrolyte. Through the imposition of a direct current (DC) upon the system, electrons traverse from the negative terminal of the DC power source towards the cathode. Here, they are absorbed by hydrogen ions (protons), thereby engendering the formation of hydrogen atoms. In the overarching scheme of water electrolysis, hydroxide and hydrogen ions migrate towards the anode and cathode respectively, a diaphragm serving as a delineating barrier between these two segments. Additionally, the hydrogen and oxygen produced at the cathode and anode, respectively, are captured by gas collectors[14].

Several electrolyte systems have been developed for water electrolysis, including alkaline water electrolysis (AWE), proton exchange membranes (PEMs), alkaline anion exchange membranes (AEMs), and solid oxide water electrolysis (SOE). Despite the utilization of diverse materials and operational parameters, these systems adhere to a common set of fundamental principles. Moreover, water electrolysis can be conducted across a spectrum of temperatures, contingent on the specific operational criteria and temperature range selected[16].

Within the realm of designing experiments for cells, a pivotal stage involves scrutinizing the experimental data to discern the optimal values for an array of parameters influencing cell performance. In this endeavor, the novel concepts of Artificial Intelligence (AI), Internet of Things (IoT), Data Science (DS), and Machine Learning (ML) emerge as relatively recent paradigms. They hold the potential to enhance the efficiency of fuel cells and augment hydrogen generation through the assimilation of historical data and predicted futures[17,18]. This study centered on examining the introduction and discussion of proton exchange membrane (PEM) electrolyzers, with a specific focus on integrating data science and machine learning principles. In pursuit of this goal, a data science procedure was devised, utilizing machine-learning algorithms and incorporating anode side catalyst parameters. The outcomes of this analysis were subsequently scrutinized using Jupyter notebook, a programming platform utilizing Python 3.9.0 facilitated by the Anaconda platform. Finally, the various models were compared, and the resulting data were visualized based on different values of model evaluation parameters which resulted in realizing the application of data science as an auxiliary tool in analyzing the data and obtaining practical models

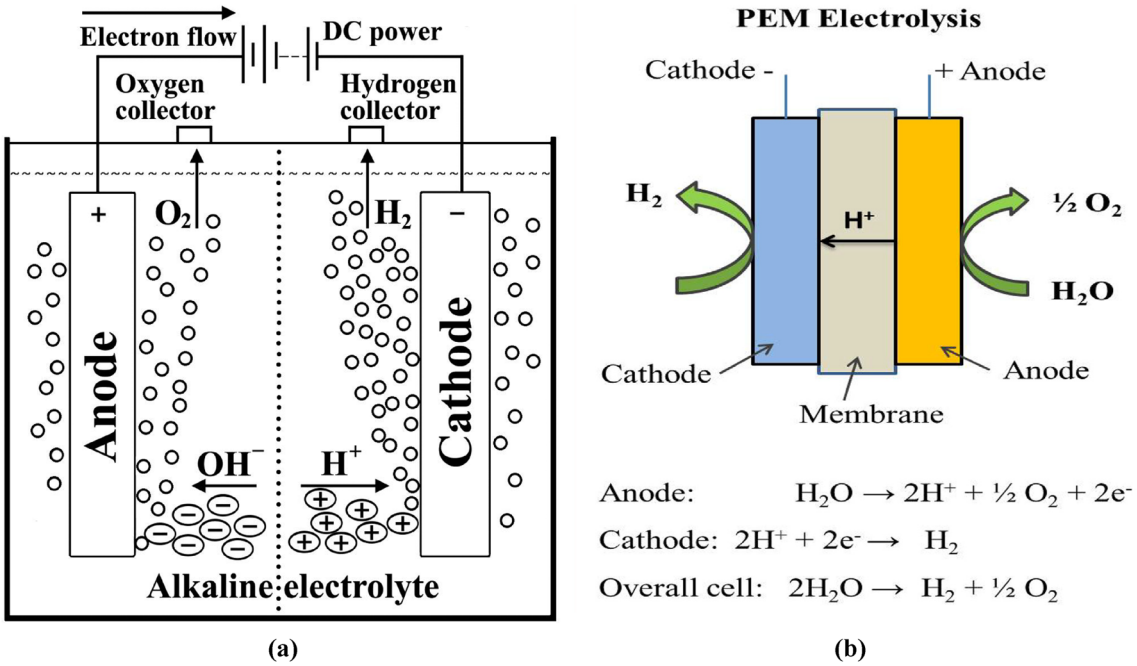


Figure 1. **(a)** Scheme principle for electrolysis cell[14] and **(b)** PEM water electrolysis.

for predicting and discovering optimal data based on the changes of other influencing experimental parameters related to the anode side of the catalyst in the production of hydrogen gas through PEM water electrolyzer.

PEM water electrolyzer (PEMWE)

The first PEM water electrolysis named as polymer electrolyte membranes or solid polymer electrolyte membranes in 1966 for U.S space applications which was related with sulfonated polystyrene ion-exchange electrolyte membrane development idealized by Grubb in 1950s to solve the problems of the alkaline water electrolysis[19,20]. In most of the PEM water electrolyzer, perfluorosulfonic acid membranes such as Nafion® and sulfonated polyetheretherketone have used as an electrolyte (proton conductor)[21,22]. These proton exchange membranes having many advantages such as lower gas permeability, high proton conductivity (0.1 ± 0.02 S cm^{-1}), lower thickness (Σ20–300 μm) and high-pressure operations. PEM water electrolysis is one of the promising methods for conversion of renewable energy to high pure hydrogen in terms of sustainability and environmental influence. Proton exchange membrane water electrolysis also offers several advantages, such as a compact design, high current density exceeding 2 $A.cm^{-2}$, high efficiency, fast response, small footprint, and operation under lower temperatures ranging from 20–80 °C. Moreover, it produces ultrapure hydrogen and oxygen as a byproduct[21,23–26]. Notably, the process of balancing PEM electrolysis plants is relatively simple, which enhances its attractiveness for industrial applications.

The primary process of a PEM water electrolyzer involves the electrochemical splitting of water into hydrogen and oxygen at the cathode and anode sides, respectively. Specifically, water is introduced to the anode side, where an OER takes place, generating oxygen (O_2), protons (H +), and electrons (e-). The electrons exit from the anode through the external power circuit, which provides the driving force (cell voltage) for the reaction. The protons that are produced travel to the cathode side through a proton-conducting membrane, resulting in a hydrogen evolution reaction (HER) that combines with the electrons to produce hydrogen, as depicted in Fig. 1b[27].

The anode catalyst in proton PEMWEs has been the subject of extensive research due to the oxygen evolution reaction (OER) being the primary source of irreversibility[28]. Typically, noble metal-based electrocatalysts such as IrO_2 are utilized for the OER in PEM water electrolysis, as it is recognized as one of the most durable materials under O_2 evolution conditions in highly acidic environments[22,29]. However, this results in a higher cost compared to alkaline water electrolysis systems. Therefore, reducing production costs while maintaining high efficiency remains a significant challenge in PEM water electrolysis[22].

Data science and machine learning

Data science theory

Data science often refers to the process of leveraging modern machine learning techniques to identify insights from data[30,31]. Over the past few years, there has been a growing trend among organizations to adopt a "data centered" approach to decision-making. As a result, there has been an increase in the formation of teams consisting of data science workers who collaborate on larger datasets, more structured code pipelines, and more consequential decisions and products[32].

The demand for advanced data analytics leading to the use of machine learning and other emerging techniques can be attributed to the advent and subsequent development of technologies such as Big Data, business Intelligence, and the applications that require automation. Sandhu[33] elucidates that machine learning is a subfield of artificial intelligence that employs computerized techniques to address problems based on historical data and information, without the need for significant modifications to the core process. Artificial intelligence, on the other hand, involves the development of algorithms and other computational techniques that imbue machines with intelligence. It comprises algorithms that can reason, act, and execute tasks using protocols that are beyond the capabilities of humans.

Machine learning theory

Machine learning is a component of artificial intelligence although it endeavors to solve problems based on historical or previous examples[34]. Unlike artificial intelligence applications, machine learning involves learning of hidden patterns within the data (data mining) and subsequently using the patterns to classify or predict an event related to the problem[35]. Simply, intelligent machines depend on knowledge to sustain their functionalities and machine learning offers such a knowledge. In essence, machine learning algorithms are embedded into machines and data streams provided so that knowledge and information are extracted and fed into the system for faster and efficient management of processes. It suffices to mention that all machine learning algorithms are also artificial intelligence techniques although not all artificial intelligence methods qualify as machine learning algorithms.

Machine learning algorithms can be broadly classified as either supervised or unsupervised, although some authors may also include reinforcement learning as a distinct category, as these techniques involve learning from data to identify patterns with the goal of reacting to an environment. Nevertheless, most literature acknowledges the two major categories of supervised and unsupervised learning algorithms. The difference between these two main classes is the existence of labels in the training data subset. As outlined by Kotsiantis[36], supervised machine learning involves the utilization of predetermined output attributes in conjunction with input attributes. These algorithms strive to predict and classify the predetermined attribute, and their performance is evaluated based on metrics such as accuracy, misclassification rate, and other relevant performance measures, which are contingent on the number of correctly predicted or classified instances of the predetermined attribute. Importantly, the learning process concludes when the algorithm attains a satisfactory level of performance[37]. According to Libbrecht and Noble[34], technically, supervised algorithms perform analytical tasks first using the training data and subsequently construct contingent functions for mapping new instance of the attribute. As stated previously, the algorithms require prespecifications of maximum settings for the desired outcome and performance levels[34,37].

Machine learning methods typically require a training subset of around 66% of the data in order to achieve satisfactory results without incurring excessive computational costs[38]. Within the supervised learning paradigm, algorithms can be categorized into either classification or regression algorithms[35,36]. In contrast, unsupervised learning does not involve a target attribute and instead focuses on pattern recognition. All variables in the analysis are used as inputs, making these techniques particularly useful for clustering and association mining. According to Hofmann[39], unsupervised learning algorithms are suitable for creating the labels in the data that are subsequently used to implement supervised learning tasks. That is, unsupervised clustering algorithms identify inherent groupings within the unlabeled data and subsequently assign label to each data value[38,40]. On the other hand, unsupervised association mining algorithms tend to identify rules that accurately represent relationships between attributes. Praveena[41] asserts that supervised learning relies on prior experience or acquired patterns within the data and typically involves a defined output variable[42–46]. The input dataset is partitioned into train and test subsets, and various studies have explored the concept of training datasets based on the desired outcome[47–49]. Algorithms employing supervised learning utilize patterns within the training dataset to predict or classify an attribute within the test subset[50,51]. Multiple authors have described the workflow of supervised machine learning, and decision trees, Naïve Bayes, and Support Vector Machines are among the most commonly used algorithms[40,52–55].

Tools and systems

There exists a plethora of tools that are designed to support the work of data scientists. These include programming languages like Python or R, statistical analysis tools such as SAS[56] and SPSS[57], integrated development environments (IDEs) like Jupyter Notebook[58,59], and automated model building systems such as AutoML[60] and AutoAI[61]. Empirical studies have shed light on how data scientists utilize these tools[62–64], as well as the features that could be augmented to enhance the user experience for those working solo[65].

Jupyter Notebook[66] is a noteworthy system, which has various versions including Google Colab[67] and JupyterLab[68]. It is an integrated development environment that is specifically tailored to meet the needs of data science workflows. The graphical user interface of Jupyter Notebook supports three core functionalities, which are vital to data science work: coding, documenting a narrative, and observing execution results[69]. Additionally, the capability to effortlessly switch between code and output cells enables data scientists to rapidly iterate on their model development and testing processes[31,62].

Developing data science procedure

Data mining

This study takes into account the current density (CD), water feed rate (WFR), catalyst loading, and high-frequency resistance (HFR) of the anode side of a proton exchange membrane water electrolysis (PEMWE) system to develop a data science procedure and train machine learning models. The datasets were obtained from Fig. 2, which displays HFR vs. average pore opening diameter (APOD) at working temperatures of 35 °C and 55 °C, and CD vs. WFR at working cell potentials of 1.9 V and 2 V at 55 °C, respectively, based on two different catalyst loading levels of 0.595 and 0.085, along with the porous transport layer (PTL) material specifications of the anode

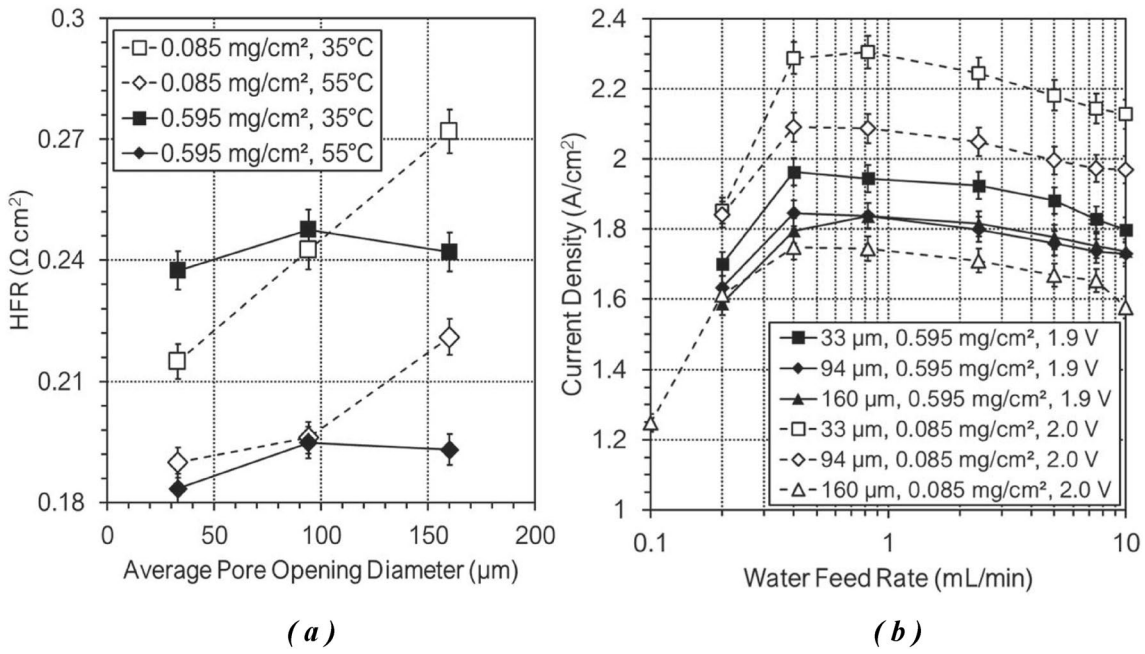


Figure 2. (**a**) trend in the HFR of the cell at 1.48 V by increasing PTL average pore opening diameter base on the IrO_2 dosage changes and (**b**) current density vs the anode water feed rate for the 4 cm^2 PEMWE cell under potentiostatic control at 55 °C different PTL average pore opening diameter base on the IrO_2 dosage changes[70].

side as the target data in the machine learning models. It should be noted that the datasets collected from Fig. 2a were only taken at 55 °C due to the temperature constraints of the CD vs. WFR experiment.

Three PTL cases have been investigated based on the average pore opening diameter, average grain diameter, areal surface porosity, average porosity and permeability as properties of these materials which are displays in Table 1.

Data generation

The datasets mentioned above were initially processed and organized using MS Excel in preparation for modeling in Python Jupyter Notebook. The Dataset-1 consisted of 42 rows and 5 columns, but this was expanded to 162 rows and 5 columns by generating random numbers within a specific range for the WFR and CD columns using the code provided below:

```
In [1]: import numpy as np
        import random
```

```
In [2]: random_list=[]
        for i in range('No.of Random Numbers'):
            random_list.append(random.random() * ('Upper Value' - 'Lower Value') + 'Lower Value')
        random_list.sort()
        for i in range (len(random_list)):
            print(round(random_list[i]))
```

The generated data have been reentered into MS Excel for sorting and initial preparing which finally has been named Dataset-2.

Input variables

In the subsequent phase, three discrete machine learning models—Regression, Support Vector Machine (SVM), and Decision Tree—will be individually employed on Dataset-1 and Dataset-2. The resultant shifts in WFR and CD concerning the PTL will be displayed in dedicated visual representations, owing to the marked distinctions in their respective datasets. To effectuate this, each dataset has been imported into Python Jupyter Notebook using the prescribed code as presented below:

```
In [2]: dataset1= pd.read_excel('c:desktop/Dataset-1.xlsx')
        dataset1.head()
```

Out[2]:

	PTL	Loading	HFR	WFR	CD
0	PTL1	0.595	0.18318	0.371324	1.70000
1	PTL1	0.595	0.18318	0.642647	1.95880
2	PTL1	0.595	0.18318	0.923897	1.94401
3	PTL1	0.595	0.18318	4.442570	1.92335
4	PTL1	0.595	0.18318	7.330800	1.87820

```
In [3]: dataset2= pd.read_excel('c:desktop/Dataset-2.xlsx')
        dataset2.head()
```

Out[3]:

	PTL	Loading	HFR	WFR	CD
0	PTL1	0.595	0.18318	0.371324	1.700000
1	PTL1	0.595	0.18318	0.382696	1.813320
2	PTL1	0.595	0.18318	0.438267	1.857506
3	PTL1	0.595	0.18318	0.497437	1.885211
4	PTL1	0.595	0.18318	0.518812	1.892424

PTL	APOD (μm)	Avg. grain dia. (μm)	Areal surface por.	Avg. por.	Permeability (m^2)
1	33	13.9	0.780	0.302	5.35e-13
2	94	30.7	0.730	0.312	1.10e-12
3	160	66.8	0.815	0.218	3.20e-13

Table 1. Properties of PTL materials[70].

Data pre-processing

The machine learning models will not work correctly in presence of the letters, so the words PTL1, PTL2 and PTL3 were replaced with a series of indexes including the numbers 0, 1 and 2 respectively by the following code:

```
In [4]: dataset1.PTL = dataset1.PTL.apply(['PTL1','PTL2','PTL3'].index)
        dataset1.head()
```

Out[4]:

	PTL	Loading	HFR	WFR	CD
0	0	0.595	0.18318	0.371324	1.70000
1	0	0.595	0.18318	0.642647	1.95880
2	0	0.595	0.18318	0.923897	1.94401
3	0	0.595	0.18318	4.442570	1.92335
4	0	0.595	0.18318	7.330800	1.87820

```
In [5]: dataset2.PTL = dataset2.PTL.apply(['PTL1','PTL2','PTL3'].index)
        dataset2.head()
```

Out[5]:

	PTL	Loading	HFR	WFR	CD
0	0	0.595	0.18318	0.371324	1.700000
1	0	0.595	0.18318	0.382696	1.813320
2	0	0.595	0.18318	0.438267	1.857506
3	0	0.595	0.18318	0.497437	1.885211
4	0	0.595	0.18318	0.518812	1.892424

In the following, since all three selected models fall under the category of supervised learning, it is necessary to specify the features and target values for the analysis. To accomplish this, two variables, X and Y, are defined to represent the features and target values, respectively. Subsequently, the train and test data must be randomly selected from the dataset to enable the models to function optimally. In this study, 30% of the data were allocated to the test set, while 70% of the data were designated as the training set. The following codes can be used to implement the above:

```
In [6]: X1 = dataset1.iloc[:, [1, 2, 3, 4]].values
        Y1 = dataset1.PTL.values
```

```
In [7]: X2 = dataset2.iloc[:, [1, 2, 3, 4]].values
        Y2 = dataset2.PTL.values
```

```
In [8]: xtrain1, xtest1, ytrain1, ytest1 = train_test_split(X1, Y1, test_size = 0.30, random_state = 1)
```

```
In [9]: xtrain2, xtest2, ytrain2, ytest2 = train_test_split(X2, Y2, test_size = 0.30, random_state = 1)
```

The acquired dataset encompasses features of diverse dimensions, collectively exerting a detrimental influence on the modeling of datasets, particularly in terms of accuracy rates, among other factors. Consequently, prior to executing the models, it is imperative to standardize the feature values utilizing the ensuing code. This procedure aims to adjust appropriately scaled dimensions conducive to effective model training.

```
In [10]: %%capture
         sc = StandardScaler()
         xtrain1 = sc.fit_transform(xtrain1)
         xtest1  = sc.transform(xtest1)
```

```
In [11]: %%capture
         sc = StandardScaler()
         xtrain2 = sc.fit_transform(xtrain2)
         xtest2  = sc.transform(xtest2)
```

Machine learning algorithms

In the following, the theory related to each of these models will be briefly explained and how to set them up along with the corresponding codes will be displayed:

Regression model

Linear regression is one of the simplest supervised learning algorithms in our toolkit. If you have ever taken an introductory statistics course in college, likely the final topic you covered was linear regression. In fact, it is so simple that it is sometimes not considered machine learning at all! Whatever you believe, the fact is that linear regression and its extensions continues to be a common and useful method of making predictions when the target vector is a quantitative value (e.g., home price, age)[71].

In this section, creating and fitting the Linear Regression model is explained which we can find the linear relationship between features and target vector besides the codes related to the prediction of the target values from the test features values:

```
In [12]: LinearRegression = LinearRegression()

In [13]: LinearRegression.fit(xtrain1, ytrain1)
Out[13]: LinearRegression()

In [14]: LinearRegression.fit(xtrain2, ytrain2)
Out[14]: LinearRegression()

In [15]: LinearRegression_ypred1= LinearRegression.predict(xtest1)

In [16]: LinearRegression_ypred2= LinearRegression.predict(xtest2)
```

Support vector machine (SVM) model

To comprehend support vector machines, it is helpful to first understand hyperplanes. In mathematical terms, a hyperplane refers to an (n-1) dimensional subspace within an n-dimensional space. Despite sounding complex, the concept is relatively simple. For instance, in a two-dimensional space, we could use a one-dimensional hyperplane (i.e., a line) to divide it. Conversely, in a three-dimensional space, a two-dimensional hyperplane (i.e., a flat plane or sheet) would suffice. In essence, a hyperplane is a generalization of this concept into n dimensions[71]. Support vector machines classify data by identifying the hyperplane that maximizes the margin between classes in the training data. In a two-dimensional example with two classes, the hyperplane is the widest straight "band" (i.e., line with margins) that separates the classes[71].

In this section, we explain the process of building and training a Support Vector Machine (SVM) model, which seeks to identify the hyperplane that maximizes the margin between classes in the training data. We also provide the code employed to predict target values from test feature values.

```
In [27]: SVM = SVC(kernel = 'linear', random_state = 0)

In [28]: SVM.fit(xtrain1, ytrain1)
Out[28]: SVC(kernel='linear', random_state=0)

In [29]: SVM.fit(xtrain2, ytrain2)
Out[29]: SVC(kernel='linear', random_state=0)

In [30]: SVM_ypred1= SVM.predict(xtest1)

In [31]: SVM_ypred2= SVM.predict(xtest2)
```

Decision tree classifier model

Tree-based learning algorithms are a broad and popular family of related nonparametric, supervised methods for both classification and regression. The basis of tree-based learners is the decision tree wherein a series of decision rules (e.g., “If their gender is male…”) are chained. The result looks vaguely like an upside-down tree, with the first decision rule at the top and subsequent decision rules spreading out below. In a decision tree, every decision rule occurs at a decision node, with the rule creating branches leading to new nodes. A branch without a decision rule at the end is called a leaf[71].

One of the primary reasons for the widespread adoption of tree-based models is their interpretability. Decision trees can be visually depicted in their entirety, thus enabling the creation of an intuitive model. This simple tree structure has spawned numerous extensions, ranging from random forests to stacking techniques[71].

In this section, the process of building and training a Decision Tree Classifier model with a maximum unit depth of 1, 2, 3, and 4 were described. This model allows us to identify decision rules based on a non-parametric relationship between features and the target vector. Furthermore, the code used to predict target values from test feature values were provided as:

Decision tree model (max-depth = 1)

```
In [42]: decisiontree1 = DecisionTreeClassifier(max_depth= 1,random_state=0)

In [43]: decisiontree1.fit(xtrain1, ytrain1)
Out[43]: DecisionTreeClassifier(max_depth=1, random_state=0)

In [44]: decisiontree1.fit(xtrain2, ytrain2)
Out[44]: DecisionTreeClassifier(max_depth=1, random_state=0)

In [45]: decisiontree1_ypred1= decisiontree1.predict(xtest1)

In [46]: decisiontree1_ypred2= decisiontree1.predict(xtest2)
```

Decision tree model (max-depth = 2)

```
In [57]: decisiontree2 = DecisionTreeClassifier(max_depth= 2,random_state=0)

In [58]: decisiontree2.fit(xtrain1, ytrain1)
Out[58]: DecisionTreeClassifier(max_depth=2, random_state=0)

In [59]: decisiontree2.fit(xtrain2, ytrain2)
Out[59]: DecisionTreeClassifier(max_depth=2, random_state=0)

In [60]: decisiontree2_ypred1= decisiontree2.predict(xtest1)

In [61]: decisiontree2_ypred2= decisiontree2.predict(xtest2)
```

Decision tree model (max-depth = 3)

```
In [72]: decisiontree3 = DecisionTreeClassifier(max_depth= 3,random_state=0)

In [73]: decisiontree3.fit(xtrain1, ytrain1)
Out[73]: DecisionTreeClassifier(max_depth=3, random_state=0)

In [74]: decisiontree3.fit(xtrain2, ytrain2)
Out[74]: DecisionTreeClassifier(max_depth=3, random_state=0)

In [75]: decisiontree3_ypred1= decisiontree3.predict(xtest1)

In [76]: decisiontree3_ypred2= decisiontree3.predict(xtest2)
```

Decision tree model (max-depth = 4)

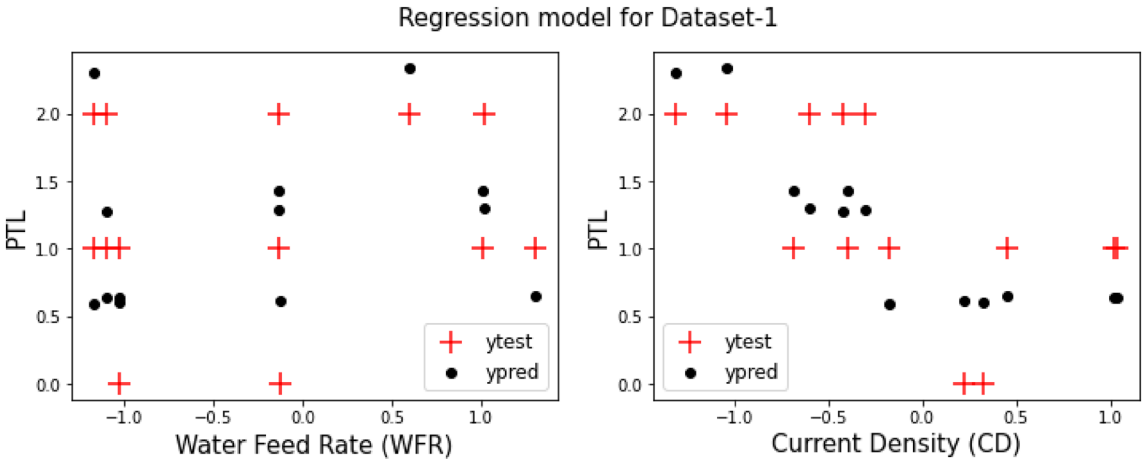
```
In [87]: decisiontree4 = DecisionTreeClassifier(max_depth= 4,random_state=0)

In [88]: decisiontree4.fit(xtrain1, ytrain1)
Out[88]: DecisionTreeClassifier(max_depth=4, random_state=0)

In [89]: decisiontree4.fit(xtrain2, ytrain2)
Out[89]: DecisionTreeClassifier(max_depth=4, random_state=0)

In [90]: decisiontree4_ypred1= decisiontree4.predict(xtest1)

In [91]: decisiontree4_ypred2= decisiontree4.predict(xtest2)
```

Model evaluation and visualization

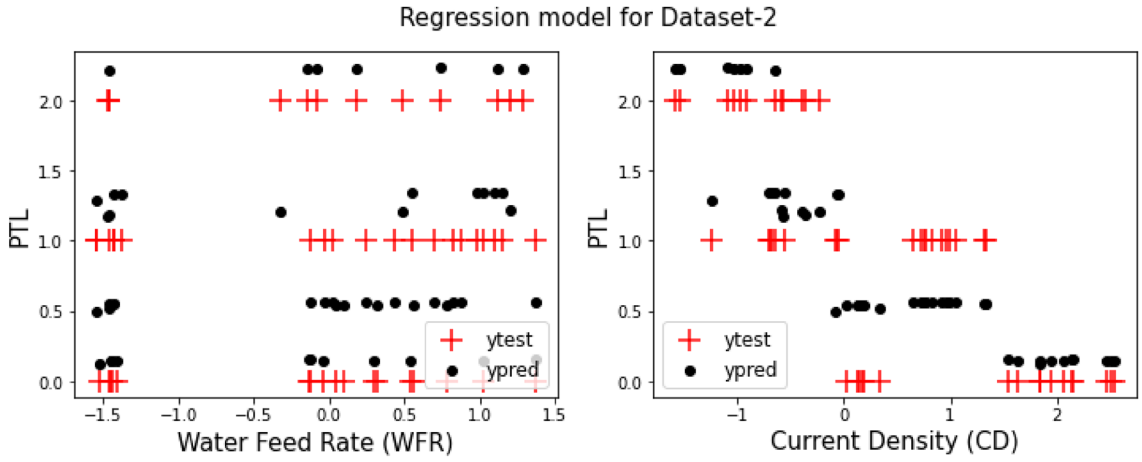
As it said in previous sections, a comparison between the prediction target values from training the models and the test target values for both Dataset-1 and Dataset-2 have been displays in Figs. 3, 4, 5, 6, 7 and 8 based on the distribution of WFR and CD vs. PTL cases.

To check the efficiency of the models, there are number of parameters (Metrics), including model score (Accuracy), mean absolute error (MAE), mean squared error (MSE) and R^2, which can be used to comparing the values of these parameters separately for each of the models using the following codes:

```
In [17]: LinearRegression_score1= round(LinearRegression.score(xtrain1, ytrain1), 2)
         LinearRegression_score1
Out[17]: 0.71

In [18]: LinearRegression_score2= round(LinearRegression.score(xtrain2, ytrain2), 2)
         LinearRegression_score2
Out[18]: 0.66

In [19]: MAE_LinearRegression_1= round(mean_absolute_error(ytest1,
                                                           LinearRegression_ypred1), 2)
         MAE_LinearRegression_1
Out[19]: 0.49

In [20]: MAE_LinearRegression_2= round(mean_absolute_error(ytest2,
                                                           LinearRegression_ypred2), 2)
         MAE_LinearRegression_2
Out[20]: 0.38

In [21]: MSE_LinearRegression_1= round(mean_squared_error(ytest1,LinearRegression_ypred1), 2)
         MSE_LinearRegression_1
Out[21]: 0.26

In [22]: MSE_LinearRegression_2= round(mean_squared_error(ytest2,LinearRegression_ypred2), 2)
         MSE_LinearRegression_2
Out[22]: 0.18

In [23]: r2_LinearRegression_1= round(r2_score(ytest1,LinearRegression_ypred1), 2)
         r2_LinearRegression_1
Out[23]: 0.46

In [24]: r2_LinearRegression_2= round(r2_score(ytest2,LinearRegression_ypred2), 2)
         r2_LinearRegression_2
Out[24]: 0.69

In [32]: SVM_score1= round(SVM.score(xtrain1, ytrain1), 2)
         SVM_score1
Out[32]: 0.79

In [33]: SVM_score2= round(SVM.score(xtrain2, ytrain2), 2)
         SVM_score2
Out[33]: 0.82

In [34]: MAE_SVM_1= round(mean_absolute_error(ytest1, SVM_ypred1), 2)
         MAE_SVM_1
Out[34]: 0.15

In [35]: MAE_SVM_2= round(mean_absolute_error(ytest2, SVM_ypred2), 2)
         MAE_SVM_2
Out[35]: 0.16

In [36]: MSE_SVM_1= round(mean_squared_error(ytest1,SVM_ypred1), 2)
         MSE_SVM_1
Out[36]: 0.15

In [37]: MSE_SVM_2= round(mean_squared_error(ytest2,SVM_ypred2), 2)
         MSE_SVM_2
Out[37]: 0.16

In [38]: r2_SVM_1= round(r2_score(ytest1,SVM_ypred1), 2)
         r2_SVM_1
Out[38]: 0.68

In [39]: r2_SVM_2= round(r2_score(ytest2,SVM_ypred2), 2)
         r2_SVM_2
Out[39]: 0.72

In [47]: decisiontree1_score1= round(decisiontree1.score(xtrain1, ytrain1), 2)
         decisiontree1_score1
Out[47]: 0.72

In [48]: decisiontree1_score2= round(decisiontree1.score(xtrain2, ytrain2), 2)
         decisiontree1_score2
Out[48]: 0.7

In [49]: MAE_decisiontree1_1= round(mean_absolute_error(ytest1, decisiontree1_ypred1), 2)
         MAE_decisiontree1_1
Out[49]: 0.46

In [50]: MAE_decisiontree1_2= round(mean_absolute_error(ytest2, decisiontree1_ypred2), 2)
         MAE_decisiontree1_2
Out[50]: 0.41
```

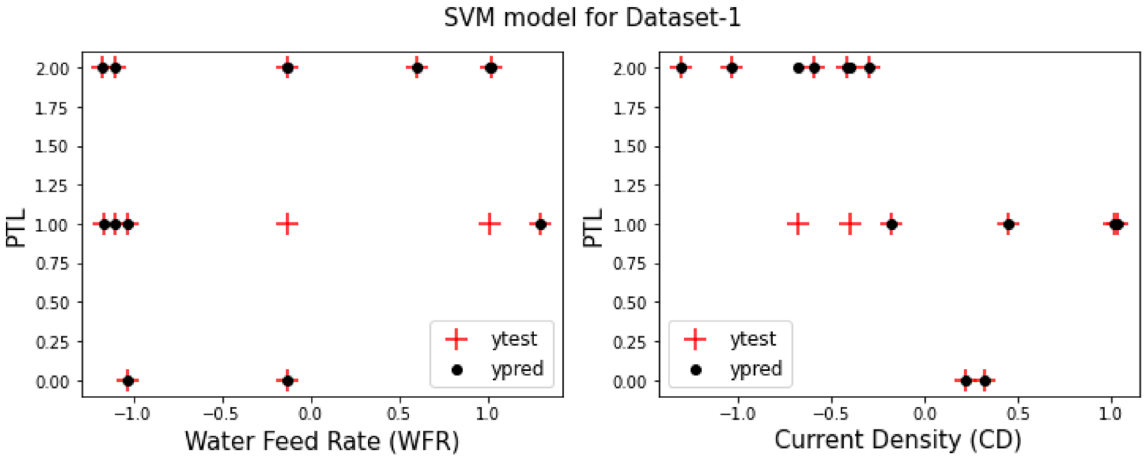


(a)

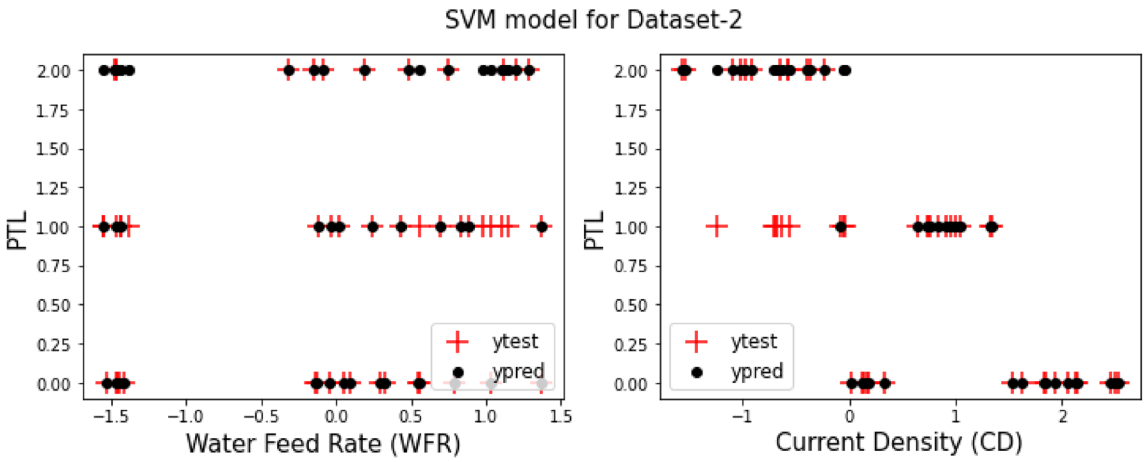


(b)

Figure 3. PTL vs. water feed rate (WFR) and current density (CD) for regression model (**a**) Dataset-1 (**b**) Dataset-2.

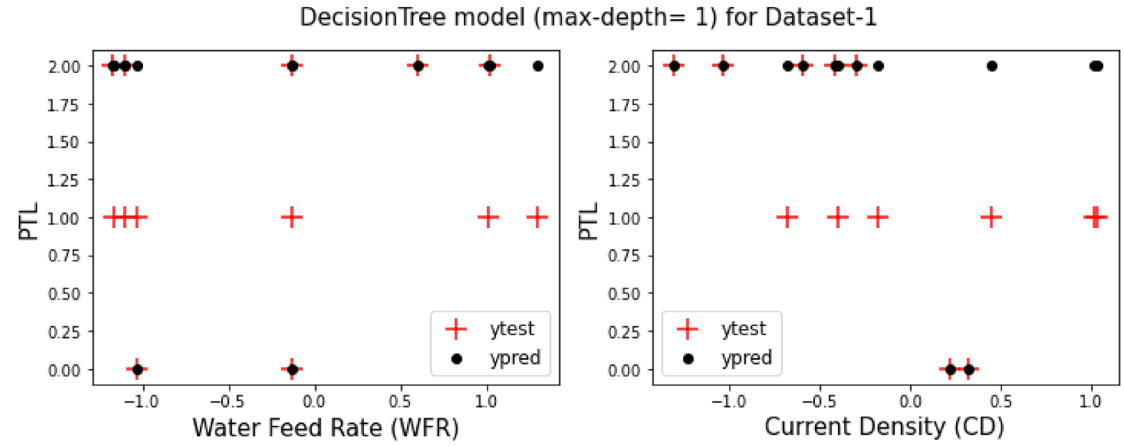


(a)

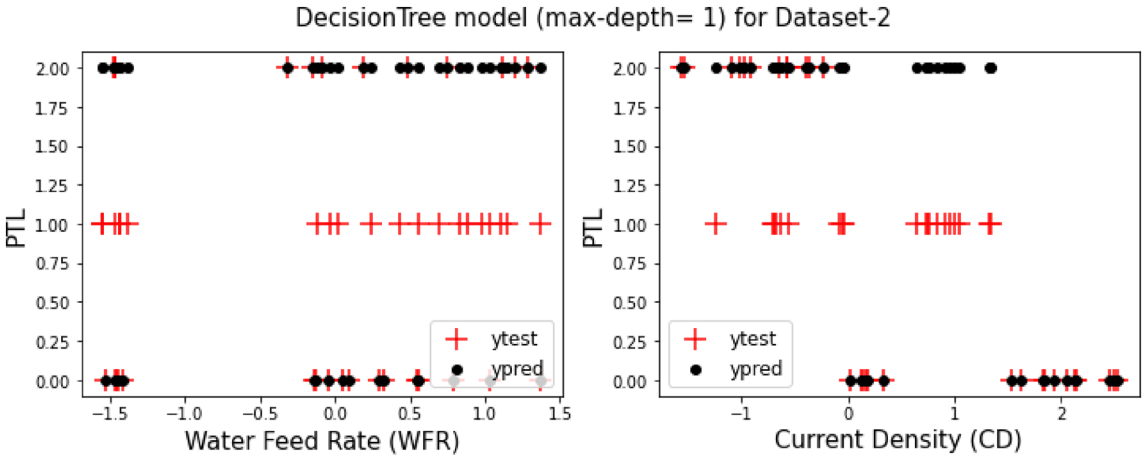


(b)

Figure 4. PTL vs. water feed rate (WFR) and current density (CD) for SVM model (**a**) Dataset-1 (**b**) Dataset-2.



(a)



(b)

Figure 5. PTL vs. water feed rate (WFR) and current density (CD) for DecisionTree (max-depth = 1) model (**a**) Dataset-1 (**b**) Dataset-2.

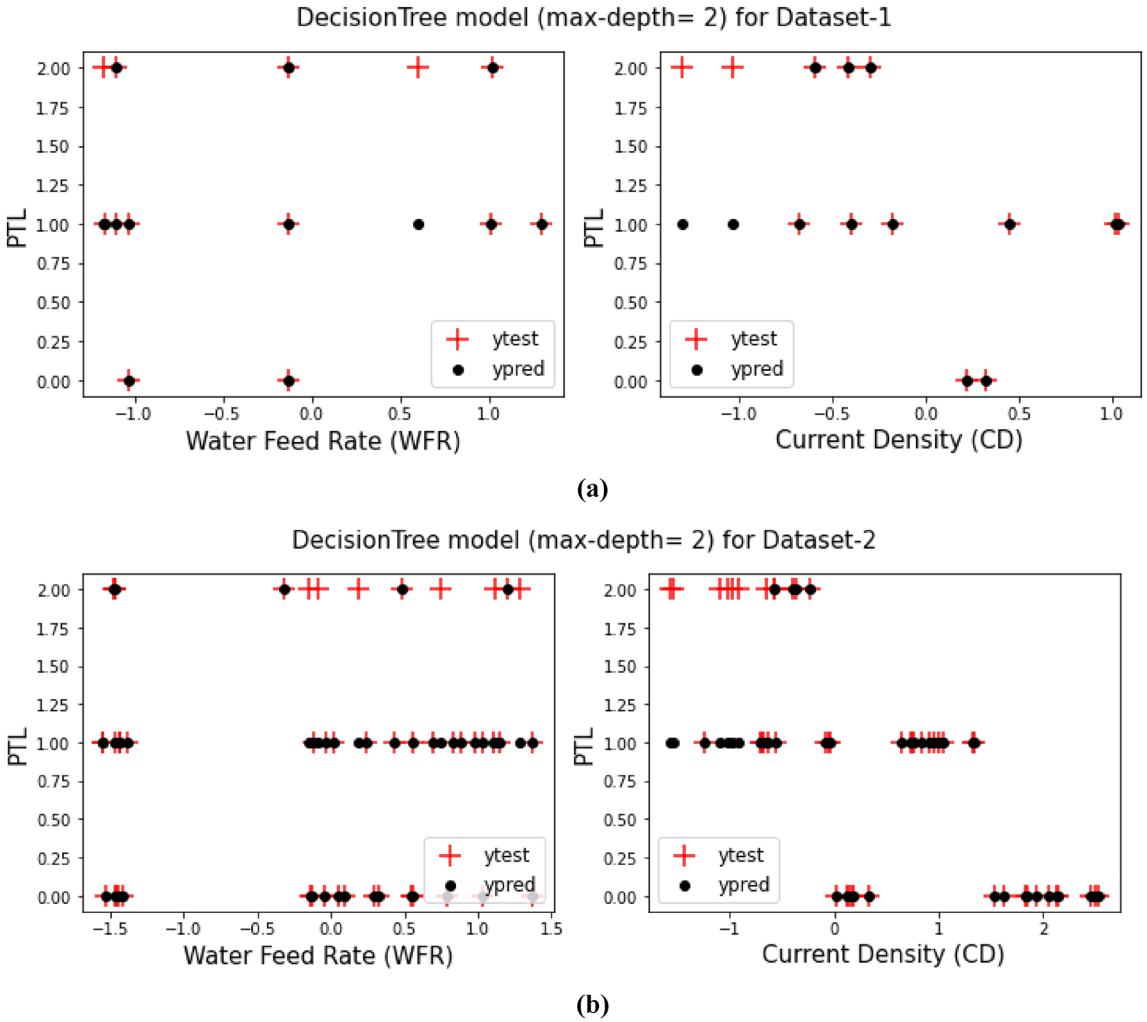
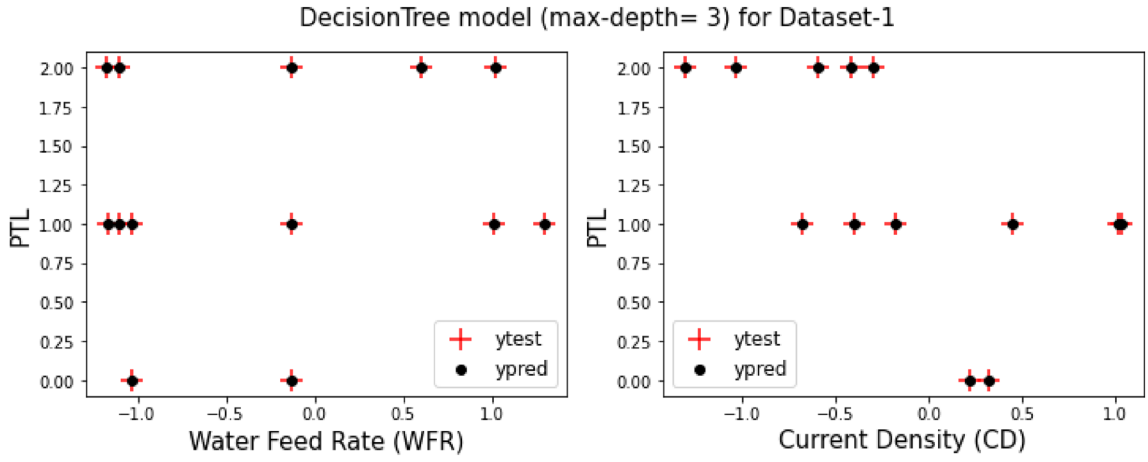
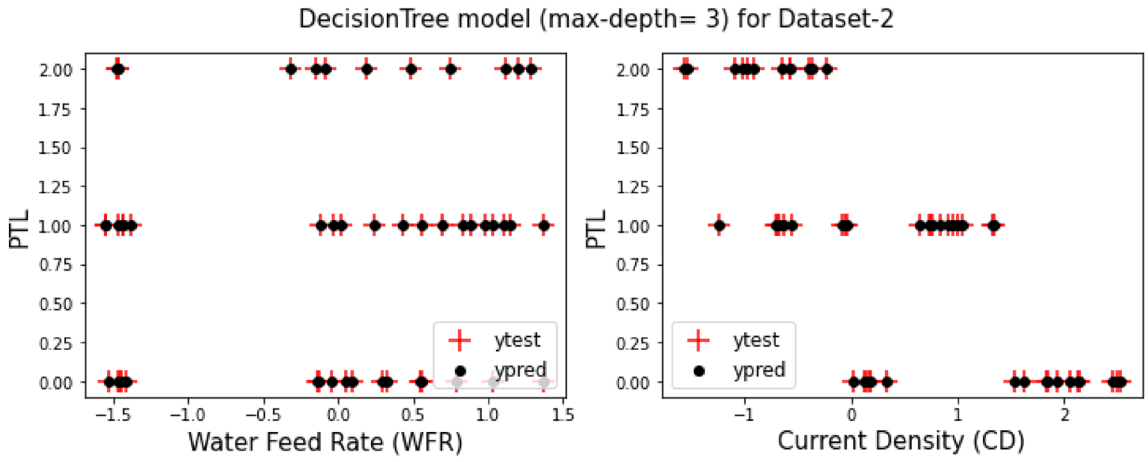


Figure 6. PTL vs. water feed rate (WFR) and current density (CD) for DecisionTree (max-depth = 2) model (**a**) Dataset-1 (**b**) Dataset-2.



(a)



(b)

Figure 7. PTL vs. water feed rate (WFR) and current density (CD) for DecisionTree (max-depth = 3) model (**a**) Dataset-1 (**b**) Dataset-2.

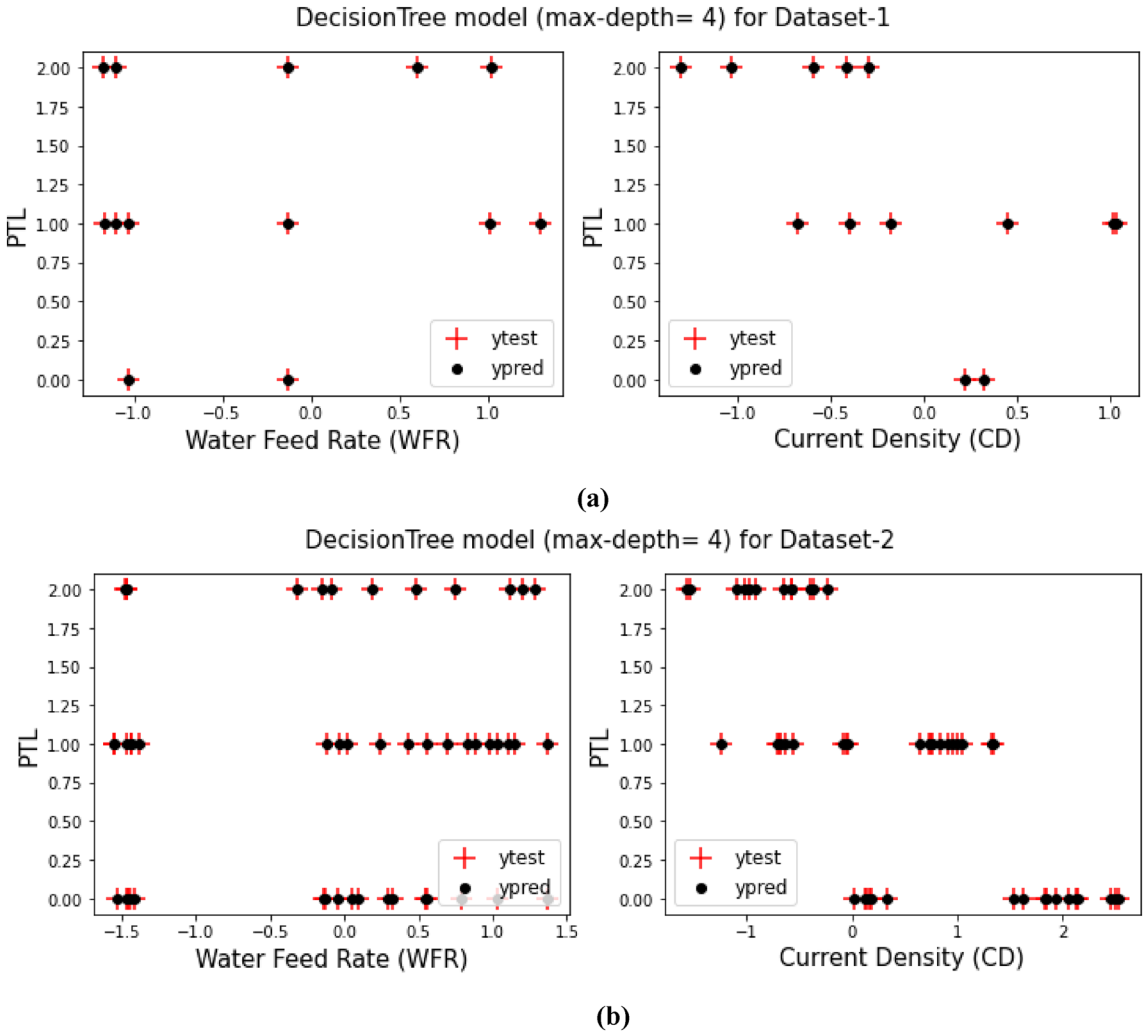


Figure 8. PTL vs. water feed rate (WFR) and current density (CD) for DecisionTree (max-depth = 4) model (**a**) Dataset-1 (**b**) Dataset-2.

```
In [51]: MSE_decisiontree1_1= round(mean_squared_error(ytest1,decisiontree1_ypred1), 2)
         MSE_decisiontree1_1
Out[51]: 0.46

In [52]: MSE_decisiontree1_2= round(mean_squared_error(ytest2,decisiontree1_ypred2), 2)
         MSE_decisiontree1_2
Out[52]: 0.41

In [53]: r2_decisiontree1_1= round(r2_score(ytest1,decisiontree1_ypred1), 2)
         r2_decisiontree1_1
Out[53]: 0.05

In [54]: r2_decisiontree1_2= round(r2_score(ytest2,decisiontree1_ypred2), 2)
         r2_decisiontree1_2
Out[54]: 0.3

In [62]: decisiontree2_score1= round(decisiontree2.score(xtrain1, ytrain1), 2)
         decisiontree2_score1
Out[62]: 0.83

In [63]: decisiontree2_score2= round(decisiontree2.score(xtrain2, ytrain2), 2)
         decisiontree2_score2
Out[63]: 0.82

In [64]: MAE_decisiontree2_1= round(mean_absolute_error(ytest1, decisiontree2_ypred1), 2)
         MAE_decisiontree2_1
Out[64]: 0.15

In [65]: MAE_decisiontree2_2= round(mean_absolute_error(ytest2, decisiontree2_ypred2), 2)
         MAE_decisiontree2_2
Out[65]: 0.14

In [66]: MSE_decisiontree2_1= round(mean_squared_error(ytest1,decisiontree2_ypred1), 2)
         MSE_decisiontree2_1
Out[66]: 0.15

In [67]: MSE_decisiontree2_2= round(mean_squared_error(ytest2,decisiontree2_ypred2), 2)
         MSE_decisiontree2_2
Out[67]: 0.14

In [68]: r2_decisiontree2_1= round(r2_score(ytest1,decisiontree2_ypred1), 2)
         r2_decisiontree2_1
Out[68]: 0.68

In [69]: r2_decisiontree2_2= round(r2_score(ytest2,decisiontree2_ypred2), 2)
         r2_decisiontree2_2
Out[69]: 0.75


In [77]: decisiontree3_score1= round(decisiontree3.score(xtrain1, ytrain1), 2)
         decisiontree3_score1
Out[77]: 1.0

In [78]: decisiontree3_score2= round(decisiontree3.score(xtrain2, ytrain2), 2)
         decisiontree3_score2
Out[78]: 1.0

In [79]: MAE_decisiontree3_1= round(mean_absolute_error(ytest1, decisiontree3_ypred1), 2)
         MAE_decisiontree3_1
Out[79]: 0.0
```

```
In [80]: MAE_decisiontree3_2= round(mean_absolute_error(ytest2, decisiontree3_ypred2), 2)
         MAE_decisiontree3_2
Out[80]: 0.0

In [81]: MSE_decisiontree3_1= round(mean_squared_error(ytest1,decisiontree3_ypred1), 2)
         MSE_decisiontree3_1
Out[81]: 0.0

In [82]: MSE_decisiontree3_2= round(mean_squared_error(ytest2,decisiontree3_ypred2), 2)
         MSE_decisiontree3_2
Out[82]: 0.0

In [83]: r2_decisiontree3_1= round(r2_score(ytest1,decisiontree3_ypred1), 2)
         r2_decisiontree3_1
Out[83]: 1.0

In [84]: r2_decisiontree3_2= round(r2_score(ytest2,decisiontree3_ypred2), 2)
         r2_decisiontree3_2
Out[84]: 1.0

In [92]: decisiontree4_score1= round(decisiontree4.score(xtrain1, ytrain1), 2)
         decisiontree4_score1
Out[92]: 1.0

In [93]: decisiontree4_score2= round(decisiontree4.score(xtrain2, ytrain2), 2)
         decisiontree4_score2
Out[93]: 1.0

In [94]: MAE_decisiontree4_1= round(mean_absolute_error(ytest1, decisiontree4_ypred1), 2)
         MAE_decisiontree4_1
Out[94]: 0.0

In [95]: MAE_decisiontree4_2= round(mean_absolute_error(ytest2, decisiontree4_ypred2), 2)
         MAE_decisiontree4_2
Out[95]: 0.0

In [96]: MSE_decisiontree4_1= round(mean_squared_error(ytest1,decisiontree4_ypred1), 2)
         MSE_decisiontree4_1
Out[96]: 0.0

In [97]: MSE_decisiontree4_2= round(mean_squared_error(ytest2,decisiontree4_ypred2), 2)
         MSE_decisiontree4_2
Out[97]: 0.0

In [98]: r2_decisiontree4_1= round(r2_score(ytest1,decisiontree4_ypred1), 2)
         r2_decisiontree4_1
Out[98]: 1.0

In [99]: r2_decisiontree4_2= round(r2_score(ytest2,decisiontree4_ypred2), 2)
         r2_decisiontree4_2
Out[99]: 1.0
```

To visualize the performance of the models, a comparison has been made between all three models on datasets 1 and 2 through the following codes and the values of model score (accuracy), mean absolute error (MAE),

mean squared error (MSE) and R^2 are displayed in Figs. 9 and 10 as the result of this work and the exact values of mentioned metrics have been tabulated in the Table 2.

Figure 3 visually demonstrates that the regression model lacks accuracy and suffers from considerable error in classifying PTL types due to the incongruity between the test data (y_{test}) and predicted data (y_{pred}). As evident from Table 2, the model achieves an accuracy of only 71% with mean absolute error (MAE), mean squared error

In [102]:
```
LR= mpatches.Patch(color= 'g', label= 'LinearRegression (LR)')
SVM= mpatches.Patch(color= 'r', label= 'SupportVectorMachine (SVM)')
D1= mpatches.Patch(color= 'b', label= 'DecisionTree(max_depth=1) (D1)')
D2= mpatches.Patch(color= 'k', label= 'DecisionTree(max_depth=2) (D2)')
D3= mpatches.Patch(color= 'm', label= 'DecisionTree(max_depth=3) (D3)')
D4= mpatches.Patch(color= 'c', label= 'DecisionTree(max_depth=4) (D4)')

names= ['LR', 'SVM', 'D1', 'D2', 'D3', 'D4']
colors= ['g', 'r', 'b', 'k', 'm', 'c']
rotation= 0

plt.figure(figsize= (14, 10))
plt.suptitle('Overall Metrics for Dataset-1', fontsize= '15')
plt.subplot(221)
plt.bar(names, [LinearRegression_score1, SVM_score1,decisiontree1_score1, decisiontree2_score1,
                decisiontree3_score1, decisiontree4_score1], color= colors, width= 0.5)
plt.ylabel('Model Score', fontsize= '15')

plt.subplot(222)
plt.bar(names, [MAE_LinearRegression_1, MAE_SVM_1, MAE_decisiontree1_1, MAE_decisiontree2_1,
           MAE_decisiontree3_1, MAE_decisiontree4_1], color= colors, width= 0.5)
plt.ylabel('Mean Absolut Error', fontsize= '15')

plt.subplot(223)
plt.bar(names, [MSE_LinearRegression_1, MSE_SVM_1, MSE_decisiontree1_1, MSE_decisiontree2_1,
           MSE_decisiontree3_1, MSE_decisiontree4_1], color= colors, width= 0.5)
plt.ylabel('Mean Squared Error', fontsize= '15')

plt.subplot(224)
plt.bar(names, [r2_LinearRegression_1, r2_SVM_1, r2_decisiontree1_1, r2_decisiontree2_1,
           r2_decisiontree3_1, r2_decisiontree4_1], color= colors, width= 0.5)
plt.ylabel('R2 Score', fontsize= '15')

plt.tight_layout()
plt.legend(loc= 'upper center', handles= [LR, SVM, D1, D2, D3, D4], ncol= 3, bbox_to_anchor= (-0.05, -0.05), fontsize= '15')
```

In [103]:
```
LR= mpatches.Patch(color= 'g', label= 'LinearRegression (LR)')
SVM= mpatches.Patch(color= 'r', label= 'SupportVectorMachine (SVM)')
D1= mpatches.Patch(color= 'b', label= 'DecisionTree(max_depth=1) (D1)')
D2= mpatches.Patch(color= 'k', label= 'DecisionTree(max_depth=2) (D2)')
D3= mpatches.Patch(color= 'm', label= 'DecisionTree(max_depth=3) (D3)')
D4= mpatches.Patch(color= 'c', label= 'DecisionTree(max_depth=4) (D4)')

names= ['LR', 'SVM', 'D1', 'D2', 'D3', 'D4']
colors= ['g', 'r', 'b', 'k', 'm', 'c']
rotation= 0

plt.figure(figsize= (14, 10))
plt.suptitle('Overall Metrics for Dataset-2', fontsize= '15')
plt.subplot(221)
plt.bar(names, [LinearRegression_score2, SVM_score2,decisiontree1_score2, decisiontree2_score2,
                decisiontree3_score2, decisiontree4_score2], color= colors, width= 0.5)
plt.ylabel('Model Score', fontsize= '15')

plt.subplot(222)
plt.bar(names, [MAE_LinearRegression_2, MAE_SVM_2, MAE_decisiontree1_2, MAE_decisiontree2_2,
           MAE_decisiontree3_2, MAE_decisiontree4_2], color= colors, width= 0.5)
plt.ylabel('Mean Absolut Error', fontsize= '15')

plt.subplot(223)
plt.bar(names, [MSE_LinearRegression_2, MSE_SVM_2, MSE_decisiontree1_2, MSE_decisiontree2_2,
           MSE_decisiontree3_2, MSE_decisiontree4_2], color= colors, width= 0.5)
plt.ylabel('Mean Squared Error', fontsize= '15')

plt.subplot(224)
plt.bar(names, [r2_LinearRegression_2, r2_SVM_2, r2_decisiontree1_2, r2_decisiontree2_2,
           r2_decisiontree3_2, r2_decisiontree4_2], color= colors, width= 0.5)
plt.ylabel('R2 Score', fontsize= '15')

plt.tight_layout()
plt.legend(loc= 'upper center', handles= [LR, SVM, D1, D2, D3, D4], ncol= 3, bbox_to_anchor= (-0.05, -0.05), fontsize= '15')
```

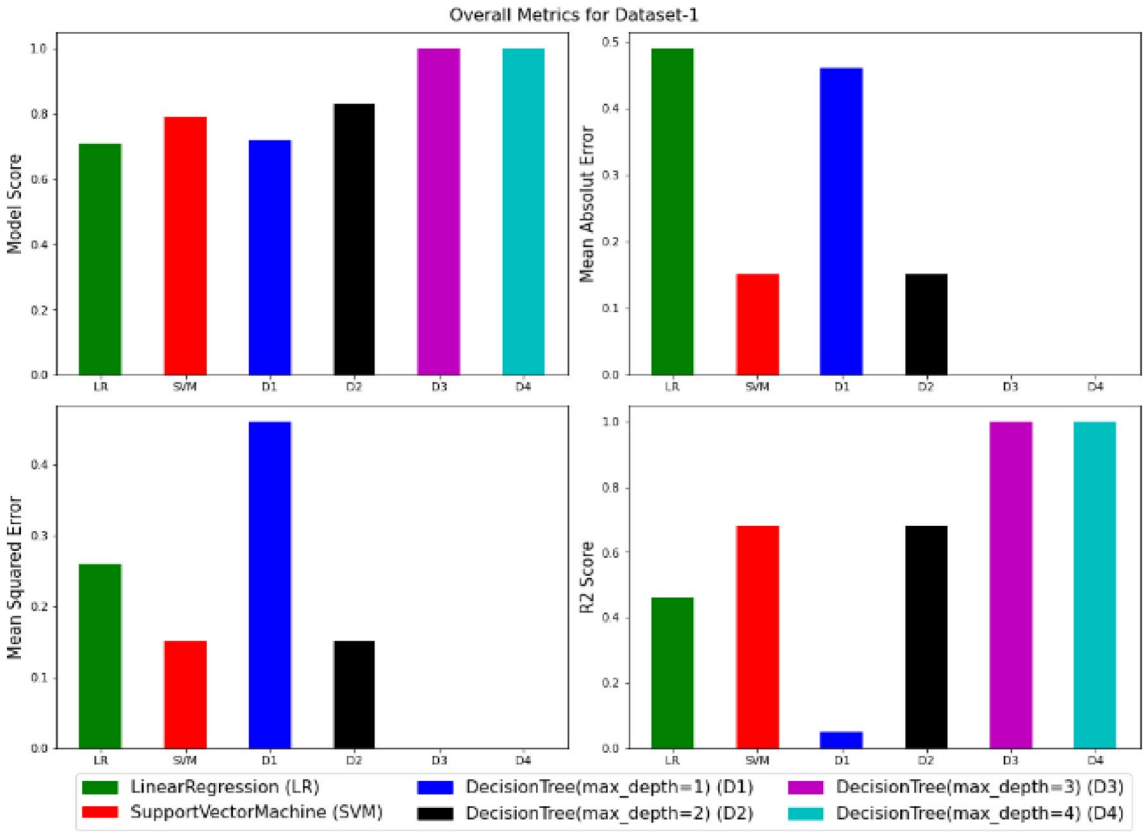


Figure 9. Overall metrics for Dataset-1.

(MSE) and R^2 of 0.49, 0.26 and 0.46, respectively. When the number of datasets increases, the accuracy declines further to 66% whereas MAE, MSE and R^2 alter to 0.38, 0.18 and 0.69, respectively. Thus, the regression model is deemed unsuitable for monitoring the system's behaviour.

Figure 4 illustrates the changes in WFR and CD for diverse PTL cases using the SVM model, ostensibly indicating that this model produces fewer prediction errors relative to the preceding model. As delineated in Table 2, the accuracy is 79% and 82% for datasets 1 and 2, respectively. The predictions from this model closely mirror the test data, thereby making the SVM model more fitting for modelling and anticipating system behaviour based on diverse parameters.

This work utilized decision tree models with maximum depths of 1, 2, 3 and 4 for modeling. Figures 5, 6, 7 and 8 illustrate the changes in WFR and CD based on PTL cases for these models. Figure 5 shows that the max-depth 1 model fails to predict PTL1 data due to its shallow structure as a result of tree pruning to a depth of one which makes the model to consider only one layer of datasets[71]. The metrics in Table 2 indicate this model is unsuitable. Figure 6 depicts the WFR and CD distribution for the max-depth 2 model, achieving 83% and 82% accuracy for datasets 1 and 2, respectively (Table 2). Unlike the previous model, it predicts all PTL cases due to its depth which could considering two layer of datasets[71], making it suitable for predicting system changes and behavior. Figures 7 and 8 show the distributions for max-depth 3 and 4 models, respectively. The test and predicted data match perfectly, indicating highly accurate performance. Table 2 elucidates that their accuracy attains a remarkable 100% for both datasets, demonstrating exemplary model training and aptitude for predicting system behavior. The more intricate structures encompass all strata of data, culminating in flawless concordance between test and predicted data. To deepen our understanding, it is imperative to recognize that in decision tree models, the target value Y is forecasted by taking into account all input feature values denoted as $X_1, X_2, \ldots, X_P$, where P signifies the number of feature values or data layers. In this scenario, a binary tree is cultivated wherein, at each node, a test is conducted on one of the inputs or layers, denoted as X_i. It is worth noting that every maximum depth of the tree corresponds to the consideration of a singular layer or feature value, denoted as X_i[72]. Consequently, in the case of PTL versus water feed rate (WFR) and current density (CD) for the Decision Tree models with maximum depths of 3 and 4, for both datasets, three and four layers of the datasets were taken into account as input values. The tree was pruned to these specified depths, resulting in elucidated models boasting an accuracy of 100%[71]. Figures 9 and 10 compare the overall metrics for all models and datasets individually, allowing model comparisons based on accuracy, MAE, MSE and R^2.

In summation, the incongruity between the test and predicted data for the regression model and its inferior predictive capability renders it an inappropriate choice for identifying PTL types. In contrast, the SVM model,

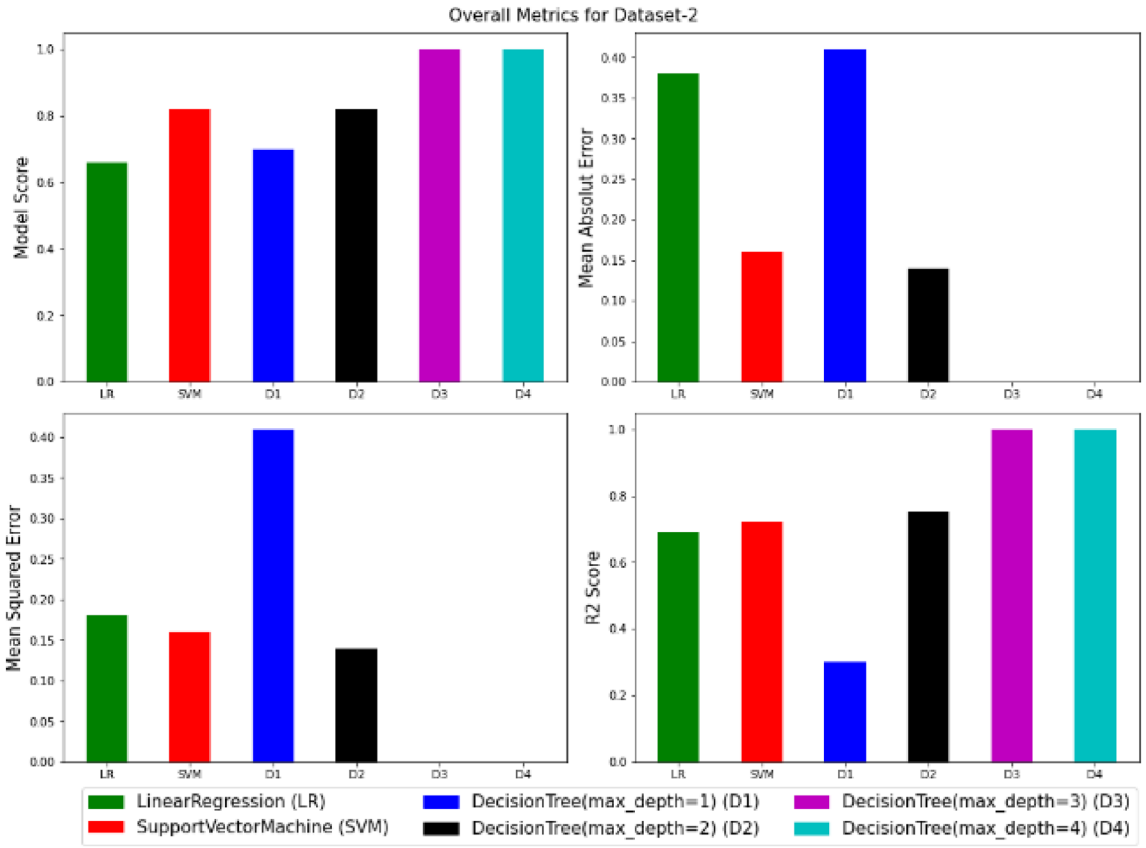


Figure 10. Overall metrics for Dataset-2.

Models	Overall metric values							
	Dataset-1				Dataset-2			
	Accuracy	MAE	MSE	R2-score	Accuracy	MAE	MSE	R2-score
Linear regression	0.71	0.49	0.26	0.46	0.66	0.38	0.18	0.69
Support vector machine	0.79	0.15	0.15	0.68	0.82	0.16	0.16	0.72
Decision tree (max-depth = 1)	0.72	0.46	0.46	0.05	0.70	0.41	0.41	0.3
Decision tree (max-depth = 2)	0.83	0.15	0.15	0.68	0.82	0.14	0.14	0.75
Decision tree (max-depth = 3)	1	0	0	1	1	0	0	1
Decision tree (max-depth = 4)	1	0	0	1	1	0	0	1

Table 2. Overall metrics values of models for Dataset-1 and Dataset-2.

due to its higher accuracy, congruous test and predicted data and comparatively lower errors, likely constitutes a more desirable option for the specified purpose.

Moreover, the shallower max-depth 1 model fails while the max-depth 2 model performs reasonably well. However, the max-depth 3 and 4 models achieve perfect accuracy and metric scores, indicating their suitability for the task due to their ability to account for all data layers through their deeper structures.

Conclusion

The integration of artificial intelligence and machine learning, with a particular focus on their ability to gather and analyze data, has become a logical and essential consideration for chemists and chemical engineers seeking to understand flow patterns, develop empirical models, and design and optimize various systems. Recent research has elucidated the crucial role of devoting significant time and resources to the various stages of data science, encompassing everything from mining and preprocessing to data generation, inspection, and visualization. Furthermore, this research has emphasized the importance of scientific expertise in leveraging the full potential of data science, particularly in the context of chemistry and chemical engineering. The data used in this study was obtained from open literature sources, with a primary focus on data science and machine learning. The project involved conducting a data science analysis on a set of data related to operational parameters affecting the anode

side of a PEM water electrolyzer for H_2 production, using various machine learning models and comparing their performance. One of the most important issues proven in this article is data generation, which clearly has a direct impact on the conclusion and analysis of the system by approaching overall metrics values to the reality. While some individuals may believe that increasing the volume of data will necessarily enhance the accuracy of models, this study demonstrates that as the number of data points increases, the models become more realistic, and the accuracy becomes more precise, even if it leads to a decrease in accuracy. As demonstrated in the text, the accuracy value for the SVM model increased after data generation, while it decreased for the regression, Decision Tree (max-depth = 1), and Decision Tree (max-depth = 2) models.

This endeavor delved into the exploration of data science and machine learning in conjunction with the hydrogen gas production via water electrolysis. It entailed an in-depth analysis of data pertaining to the anode side catalyst of PEMWE, with a particular emphasis on the criticality of data pre-processing, notably data generation. Consequently, it was substantiated that employing models for prognostication, analysis, and optimization profoundly impacts system efficiency. Presently, the substitution of non-renewable energy sources with renewable alternatives, as well as the development of novel, efficient, and cost-effective solutions for storing excess electricity generated by renewable systems like wind and solar, stands as a paramount challenge for humanity. This shift has profound implications for the future of our planet. Hence, we anticipate that this research, with its attendant benefits and forward-looking perspectives, will positively contribute to the advancement of hydrogen gas production as a clean fuel. Moreover, it holds the promise of informing the design of a storage system capable of accommodating excess electricity generated from renewable sources. This will be achieved through the meticulous analysis of data pertaining to the anode side of PEMWE and the derivation of a robust model for the precise prediction, analysis, and optimization of system efficiency. In conclusion, data science and machine learning provide valuable insights and can serve as useful tools for chemists and chemical engineers. They can expand their knowledge base and add to their toolbox.

Data availability

All data generated or analysed during this study are included in this published article.

Received: 18 August 2023; Accepted: 9 November 2023
Published online: 20 November 2023

References

1. Fan, G. *et al.* Comprehensive analysis and multi-objective optimization of a power and hydrogen production system based on a combination of flash-binary geothermal and PEM electrolyzer. *Int. J. Hydrogen Energy* **46**(68), 33718–33737. https://doi.org/10.1016/j.ijhydene.2021.07.206 (2021).
2. Hussein, A. K. Applications of nanotechnology in renewable energies - A comprehensive overview and understanding. *Renew. Sustain. Energy Rev.* **42**, 460–476. https://doi.org/10.1016/j.rser.2014.10.027 (2015).
3. Ibrahim, H., Ilinca, A. & Perron, J. Energy storage systems-Characteristics and comparisons. *Renew. Sustain. Energy Rev.* **12**(5), 1221–1250. https://doi.org/10.1016/j.rser.2007.01.023 (2008).
4. Rand, D. A. J. A journey on the electrochemical road to sustainability. *J. Solid State Electrochem.* **15**(7–8), 1579–1622. https://doi.org/10.1007/s10008-011-1410-z (2011).
5. Babic, U., Tarik, M., Schmidt, T. J. & Gubler, L. Understanding the effects of material properties and operating conditions on component aging in polymer electrolyte water electrolyzers. *J. Power Sources* https://doi.org/10.1016/j.jpowsour.2020.227778 (2020).
6. Pourrahmani, H., Zahedi, R., Daneshgar, S. & Vanherle, J. Lab-scale investigation of the integrated backup/storage system for wind turbines using alkaline electrolyzer. *Energies* https://doi.org/10.3390/en16093761 (2023).
7. Veziroglu, A. & MacArio, R. Fuel cell vehicles: State of the art with economic and environmental concerns. *Int. J. Hydrogen Energy* **36**(1), 25–43. https://doi.org/10.1016/j.ijhydene.2010.08.145 (2011).
8. Zahedi, R., Forootan, M. M., Ahmadi, R. & Keshavarzzadeh, M. Exergy-economic assessment of a hybrid power, cooling and heating generation system based on SOFC. *Heliyon* **9**(5), e16164. https://doi.org/10.1016/j.heliyon.2023.e16164 (2023).
9. Toghyani, S., Afshari, E., Baniasadi, E., Atyabi, S. A. & Naterer, G. F. Thermal and electrochemical performance assessment of a high temperature PEM electrolyzer. *Energy* **152**, 237–246. https://doi.org/10.1016/j.energy.2018.03.140 (2018).
10. Zeng, K. & Zhang, D. Recent progress in alkaline water electrolysis for hydrogen production and applications. *Prog. Energy Combust. Sci.* **36**(3), 307–326. https://doi.org/10.1016/j.pecs.2009.11.002 (2010).
11. Mohammadi, A. & Mehrpooya, M. A comprehensive review on coupling different types of electrolyzer to renewable energy sources. *Energy* **158**, 632–655. https://doi.org/10.1016/j.energy.2018.06.073 (2018).
12. E. Zoulias, E. Varkaraki, A review on water electrolysis. *Tcjst* 4(2), 41–71, 2004, [
13. Kreuter, W. & Hofmann, H. Electrolysis: The important energy transformer in a world of sustainable energy. *Int. J. Hydrogen Energy* **23**(8), 661–666. https://doi.org/10.1016/S0360-3199(97)00109-2 (1998).
14. K. Praveen, M. Sethumadhavan, On the extension of XOR step construction for optimal contrast grey level visual cryptography. *2017 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI* 2017, https://doi.org/10.1109/ICACCI.2017.8125843.
15. Rashid, M. M., AlMesfer, M. K., Naseem, H. & Danish, M. Hydrogen production by water electrolysis: A review of alkaline water electrolysis, PEM water electrolysis and high temperature water electrolysis. *Int. J. Eng. Adv. Technol.* **3**, 2249–8958 (2015).
16. Chi, J. & Yu, H. Water electrolysis based on renewable energy for hydrogen production. *Cuihua Xuebao/Chinese J. Catal.* **39**(3), 390–394. https://doi.org/10.1016/S1872-2067(17)62949-8 (2018).
17. Pourrahmani, H. *et al.* The applications of Internet of Things in the automotive industry: A review of the batteries, fuel cells, and engines. *Internet of Things Netherlands* **19**, 100579. https://doi.org/10.1016/j.iot.2022.100579 (2022).
18. Entezari, A., Aslani, A., Zahedi, R. & Noorollahi, Y. Artificial intelligence and machine learning in energy systems: A bibliographic perspective. *Energy Strateg. Rev.* **45**, 101017. https://doi.org/10.1016/j.esr.2022.101017 (2023).
19. Grigoriev, S. A., Fateev, V. N., Bessarabov, D. G. & Millet, P. Current status, research trends, and challenges in water electrolysis science and technology. *Int. J. Hydrogen Energy* **45**(49), 26036–26058. https://doi.org/10.1016/j.ijhydene.2020.03.109 (2020).
20. M. A. Khan *et al.*, *Recent Progresses in Electrocatalysts for Water Electrolysis*, vol. 1, no. 4. Springer Singapore, 2018. https://doi.org/10.1007/s41918-018-0014-z.
21. Ju, H. K., Badwal, S. & Giddey, S. A comprehensive review of carbon and hydrocarbon assisted water electrolysis for hydrogen production. *Appl. Energy* **231**(May), 502–533. https://doi.org/10.1016/j.apenergy.2018.09.125 (2018).

22. Xu, W. & Scott, K. The effects of ionomer content on PEM water electrolyser membrane electrode assembly performance. *Int. J. Hydrogen Energy* **35**(21), 12029–12037. https://doi.org/10.1016/j.ijhydene.2010.08.055 (2010).
23. Nikolaidis, P. & Poullikkas, A. A comparative overview of hydrogen production processes. *Renew. Sustain. Energy Rev.* **67**, 597–611. https://doi.org/10.1016/j.rser.2016.09.044 (2017).
24. Millet, P. *et al.* PEM water electrolyzers: From electrocatalysis to stack development. *Int. J. Hydrogen Energy* **35**(10), 5043–5052. https://doi.org/10.1016/j.ijhydene.2009.09.015 (2010).
25. Grigoriev, S. A., Millet, P. & Fateev, V. N. Evaluation of carbon-supported Pt and Pd nanoparticles for the hydrogen evolution reaction in PEM water electrolysers. *J. Power Sources* **177**(2), 281–285. https://doi.org/10.1016/j.jpowsour.2007.11.072 (2008).
26. Grigoriev, S. A., Porembsky, V. I. & Fateev, V. N. Pure hydrogen production by PEM electrolysis for hydrogen energy. *Int. J. Hydrogen Energy* **31**(2), 171–175. https://doi.org/10.1016/j.ijhydene.2005.04.038 (2006).
27. ShivaKumar, S. & Himabindu, V. Hydrogen production by PEM water electrolysis – A review. *Mater. Sci. Energy Technol.* **2**(3), 442–454. https://doi.org/10.1016/j.mset.2019.03.002 (2019).
28. Rozain, C., Mayousse, E., Guillet, N. & Millet, P. Influence of iridium oxide loadings on the performance of PEM water electrolysis cells: Part II - Advanced oxygen electrodes. *Appl. Catal. B Environ.* **182**, 123–131. https://doi.org/10.1016/j.apcatb.2015.09.011 (2016).
29. Siracusano, S., Baglio, V., Moukheiber, E., Merlo, L. & Arico, A. S. Performance of a PEM water electrolyser combining an IrRu-oxide anode electrocatalyst and a shortside chain Aquivion membrane. *Int. J. Hydrogen Energy* **40**(42), 14430–14435. https://doi.org/10.1016/j.ijhydene.2015.04.159 (2015).
30. Kim, M., Zimmermann, T., DeLine, R. & Begel, A. The emerging role of data scientists on software development teams. *Proc. - Int. Conf. Softw. Eng.* **14–22**, 96–107. https://doi.org/10.1145/2884781.2884783 (2016).
31. Muller, M. *et al.* How data science workers work with data. *Conf. Hum. Factors Comput. Syst. Proc.* https://doi.org/10.1145/3290605.3300356 (2019).
32. Zhang, A. X., Muller, M. & Wang, D. How do data science workers collaborate? Roles, workflows, and tools. *Proc. ACM Human-Computer Interact.* **4**(CSCW1), 1–23. https://doi.org/10.1145/3392826 (2020).
33. "Sandhu, T."
34. Libbrecht, M. W. & Noble, W. S. Machine learning applications in genetics and genomics. *Nat. Rev. Genet.* **16**(6), 321–332. https://doi.org/10.1038/nrg3920 (2015).
35. A. Ethem, "Introduction to Machine Learning - Ethem Alpaydin - Google Books," *Massachusetts Institute of Technology*. 2020.
36. Kotsiantis, S. B., Zaharakis, I. & Pintelas, P. Supervised machine learning: A review of classification techniques. *Emerg. Artif. Intell. Appl. Comput. Eng.* **160**(1), 3–24 (2007).
37. "MathWorks."
38. M. W. Berry, *[Unsupervised and Semi-Supervised Learning] Michael W. Berry, Azlinah Mohamed, Bee Wah Yap - Supervised and Unsupervised Learning for Data Science (2020, Springer International Publishing) - libgen.lc.pdf*.
39. Hofmann, T. Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn.* **42**(1–2), 177–196. https://doi.org/10.1023/A:1007617005950 (2001).
40. J. Dougherty, R. Kohavi, M. Sahami, *Supervised and Unsupervised Discretization of Continuous Features*. Morgan Kaufmann Publishers, Inc., 1995. doi: https://doi.org/10.1016/b978-1-55860-377-6.50032-3.
41. Praveena, M. & Jaiganesh, V. A literature review on supervised machine learning algorithms and boosting process. *Int. J. Comput. Appl.* **169**(8), 32–35. https://doi.org/10.5120/ijca2017914816 (2017).
42. Deo, R. C. Machine learning in medicine. *Circulation* **132**(20), 1920–1930. https://doi.org/10.1161/CIRCULATIONAHA.115.001593 (2015).
43. Wu, M. J. *et al.* Identification and individualized prediction of clinical phenotypes in bipolar disorders using neurocognitive data, neuroimaging scans and machine learning. *Neuroimage* **145**, 254–264. https://doi.org/10.1016/j.neuroimage.2016.02.016 (2017).
44. Palma, S. I. C. J. *et al.* Machine learning for the meta-analyses of microbial pathogens' volatile signatures. *Sci. Rep.* **8**(1), 1–15. https://doi.org/10.1038/s41598-018-21544-1 (2018).
45. Oudah, M. & Henschel, A. Taxonomy-aware feature engineering for microbiome classification. *BMC Bioinformatics* **19**(1), 1–13. https://doi.org/10.1186/s12859-018-2205-3 (2018).
46. Mullainathan, S. & Spiess, J. Machine learning: An applied econometric approach. *J. Econ. Perspect.* **31**(2), 87–106. https://doi.org/10.1257/jep.31.2.87 (2017).
47. Crawford, M., Khoshgoftaar, T. M., Prusa, J. D., Richter, A. N. & Al Najada, H. Survey of review spam detection using machine learning techniques. *J. Big Data* https://doi.org/10.1186/s40537-015-0029-9 (2015).
48. Jaspers, S., DeTroyer, E. & Aerts, M. Machine learning techniques for the automation of literature reviews and systematic reviews in EFSA. *EFSA Support. Publ.* https://doi.org/10.2903/sp.efsa.2018.en-1427 (2018).
49. Dinov, I. D. Methodological challenges and analytic opportunities for modeling and interpreting Big Healthcare Data. *Gigascience* **5**(1), 1–15. https://doi.org/10.1186/s13742-016-0117-6 (2016).
50. Trilling, D. & Boumans, J. W. Automatische inhoudsanalyse van Nederlandstalige data : Een overzicht en onderzoeksagenda. *Tijdschr. Voor Commun.* **46**(1), 5–24 (2018).
51. Van Nieuwenburg, E. P. L., Liu, Y. H. & Huber, S. D. Learning phase transitions by confusion. *Nat. Phys.* **13**(5), 435–439. https://doi.org/10.1038/nphys4037 (2017).
52. Dobson, J. E. Can an algorithm be disturbed? Machine learning, intrinsic criticism, and the digital humanities. *College Literat.* **42**(4), 543–564. https://doi.org/10.1353/lit.2015.0037 (2015).
53. Downing, N. S. *et al.* Describing the performance of U.S. hospitals by applying big data analytics. *PLoS One* **12**(6), 1–14. https://doi.org/10.1371/journal.pone.0179603 (2017).
54. Hoang, X. D. & Nguyen, Q. C. Botnet detection based on machine learning techniques using DNS query data. *Futur. Internet* **10**(5), 1–11. https://doi.org/10.3390/FI10050043 (2018).
55. Kothari, U. C. & Momayez, M. Machine learning: A novel approach to predicting slope instabilities. *Int. J. Geophys.* https://doi.org/10.1155/2018/4861254 (2018).
56. "RC Littell, WW Stroup, GA Milliken, RD Wolfinger,."
57. Mikut, R. & Reischl, M. Data mining tools. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **1**(5), 431–443 (2011).
58. Brian Granger, Chris Colbert, and Ian Rose.
59. Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez.
60. S. Liu *et al.*, An ADMM based framework for AutoML pipeline configuration. In *AAAI 2020 - 34th AAAI Conf. Artif. Intell.*, 4892–4899, 2020, https://doi.org/10.1609/aaai.v34i04.5926.
61. Wang, D. *et al.* Human-AI collaboration in data science. *Proc. ACM Human-Computer Interact.* **3**, 1–24. https://doi.org/10.1145/3359313 (2019).
62. Kery, M. B., Radensky, M., Arya, M., John, B. E. & Myers, B. A. The story in the notebook: Exploratory data science using a literate programming tool. *Conf. Hum. Factors Comput. Syst. Proc.* https://doi.org/10.1145/3173574.3173748 (2018).
63. Kery, M. B., John, B. E., O'Flaherty, P., Horvath, A. & Myers, B. A. Towards effective foraging by data scientists to find past analysis choices. *Conf. Hum. Factors Comput. Syst. Proc.* https://doi.org/10.1145/3290605.3300322 (2019).

64. Rule, A., Tabard, A. & Hollan, J. D. Exploration and explanation in Computational notebooks. *Conf. Hum. Factors Comput. Syst. Proc.* https://doi.org/10.1145/3173574.3173606 (2018).
65. Wang, A. Y., Mittal, A., Brooks, C. & Oney, S. How data scientists use computational notebooks for real-time collaboration. *Proc. ACM Hum. Comput. Interact.* https://doi.org/10.1145/3359141 (2019).
66. T. Kluyver *et al.*, Project Jupyter | Home. *Jupyter Notebooks -- a publishing format for reproducible computational workflows*. pp. 87–90, 2016. [Online]. Available: https://jupyter.org/
67. Google Colab, Welcome to colaboratory – colaboratory. *Getting Started - Introduction*. p. 1, 2022. https://colab.research.google.com/notebooks/intro.ipynb#recent=true%0Ahttps://colab.research.google.com/
68. GitHub - jupyterlab/jupyterlab: JupyterLab computational environment. https://github.com/jupyterlab/jupyterlab
69. S. Kross, P. J. Guo, Practitioners teaching data science in industry and academia. pp. 1–14, 2019, doi: https://doi.org/10.1145/3290605.3300493.
70. Lopata, J. *et al.* Effects of the transport/catalyst layer interface and catalyst loading on mass and charge transport phenomena in polymer electrolyte membrane water electrolysis devices. *J. Electrochem. Soc.* **167**(6), 064507. https://doi.org/10.1149/1945-7111/ab7f87 (2020).
71. P. Joshi, *Python Machine Learning CookBook*. 2016.
72. Voyant, C. *et al.* Machine learning methods for solar radiation forecasting: A review. *Renew. Energy* **105**, 569–582. https://doi.org/10.1016/j.renene.2016.12.095 (2017).

Author contributions

Mahdi Arjmandi: Investigation, Resources, Data curation, Writing- Original draft preparation, Methodology, Software. Moslem Fattahi: Conceptualization, Writing- Reviewing and Editing, Visualization, Supervision. Mohsen Motevassel: Writing- Reviewing and Editing, Supervision. Hosna Rezaveisi: Investigation, Data curation, Methodology, Software.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to M.F.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.