A Three-Pronged Computational Approach for Evaluating Density Based Semi Empirical Equations of Supercritical Extraction Process and Data

Srinidhi[1, *]

[1]*Department of Chemical and Biological Engineering, School of Engineering and Applied Sciences, The State University of New York at Buffalo, Buffalo, NY 14260.*

[*]Corresponding author, e-mail: srinidh2@buffalo.edu ; msrinidhi2@gmail.com
ORCID: 0000-0002-5318-8639 ;

Abstract

Software programs for parameter estimation, phase visualization and predictive modeling of supercritical extraction process and data using algorithms is presented in this work. A contextually appropriate, iterative, ordinary least squares estimation and selection method is developed for estimating model coefficients of density based semi empirical model equations associated with this process and data. Visualization of the phase behaviors projected by the specific density based semiempirical model equation(s) is also performed iteratively by plotting three-dimensional surfaces involving the state variables and solute solubility mole fraction. Predictive modeling of input empirical data has been implemented using three supervised machine learning algorithms (Multilayer perceptron, K-nearest neighbors and support vector machine). Hyperparameter optimization of the machine learning algorithms is performed prior to prediction. Detailed analysis of the prediction is conducted by using standard scoring metrics and descriptive charts. Theoretical inference and discrepancies regarding the predicted window of maximum solubility, modeling efficiency, vapor liquid equilibrium and phase behaviors projected by the model equations have been elucidated from the program outputs. In summary, these programs are first of its kind, accurate, reliable and simple computational tools for evaluating / designing density based semiempirical equation(s) of supercritical extraction process and associated data.

Keywords: Parameter Estimation, Phase Visualization, Predictive modeling, Ordinary Least Squares, Machine Learning.

Introduction

Theoretical, Empirical and Semi empirical Models are being developed and studied for modeling and understanding Super/subcritical fluid extraction processes (Huang et al. 2012; Rai et al. 2014). In particular, Density based Semiempirical model equations (DBSE Model Equations) are very popular and are being designed for modeling this process and therefore is part of a growing body of research (Hawthorne 1990; Herrero et al. 2010; Knez et al. 2013; Alwi and Garlapati 2021a).

Novel DBSE models are developed with the aim to capture (approximate) and reproduce data specific non linearity and complexity (dynamic and non-dynamic behavior) in the process. Modeling in this scenario is primarily focused on the operating range of the process parameters observed during desired output/yield levels (Tabernero et al. 2010). Unfortunately, in most cases, this window (presumably rich in information) is narrow and is solute specific. Almost every study regarding a novel DBSE model have proceeded by distilling facts about the variation in solvating power observed in the process (from similar studies) and drawing fundamental relations between the operating process parameters and the dependent variable [ln(y), T, P, D]. A good and elegant example for this is the study and model presented by (Asgarpour Khansary et al. 2015). Least squares modeling is a subclass of Black box modeling and has been extensively employed for estimating model parameters, their confidence regions (Bounds/Intervals) and importantly for identifying causation of variance in linear models. Herein, Ordinary least squares estimation method is used for estimating parameter coefficients (and their confidence regions) present in DBSE Model equations (Lakshmi et al. 2021).

Further, A necessary requirement for the design of DBSE models is the qualitative and quantitative knowledge of phase behavior of components in the reaction mix during the process. Phase diagrams illustrate important differentials in vapor pressure curves of pure CO2 and other reaction components in the presence of solutes. This information is crucial for accurately identifying operating conditions wherein melting of the reaction mix leading to a desirable solute rich liquid phase occurs. In essence, phase diagrams are central to the process of finding regions (boundaries) of importance in the P-T-D-x (Pressure-Temperature-Density-solute solubility mole fraction) projections, wherein separations and extraction is actually possible and occurs in reality (Bartle et al. 1991). These regions (phase boundaries) depict equilibrium planes and latency of reaction mix that aid in process design and this is considered as a multifaceted and multi-attribute dependent endeavour. These attributes can be and are not limited to,

1. Regions where solvent compression occurs leading to repulsive solute-solvent interactions causing undesired immiscibility.
2. Regions where two-phase retrograde condensation/crystallization occurs near the lower and upper crossover regions/planes/edges.
3. Regions (edges/paths/points/trajectories) depicting the component(s) latency (phase change), chemical potential thermal stability of the solute leading to variations in solvating power/effect. Physical properties of solutes vary widely and significantly amount to differences during solute solubility prediction.

Machine learning algorithms, in recent years, are gaining importance and are being developed for predictive modeling at an accelerating pace for engineering applications. ML algorithms can accommodate (consider) ‘n’ number of parameters, and still can predictively model processes with desired tolerance, precision and accuracy. Invaluable for accountability and research applications, hyperparameters associated with ML algorithms offers the choice of model optimization and validation. Standardized ML algorithms are applied to model a multitude of phenomena/processes in Engineering (Selvaratnam and Koodali 2021). Therefore, with the fast parametrization and modeling of analytical and industrial processes, supervised learning models like, Regression, Multilayer Perceptron, Support Vector Machine and K-nearest neighbours are (can also be) specially applied to these processes. For Chemistry and Chemical Engineering applications, A number of Software program packages based on supervised learning are already available and are always under continuous development (Khatib and de Jong 2020). In recent years, estimating/predicting solute solubility during the supercritical fluid extraction process is gaining importance and necessitates predictive modeling of this process (Butler et al. 2018; Schweidtmann et al. 2021; Roach et al. 2023). The reliable and utilitarian software program can be used to accurately describe extant pattern and behavior in the measured data associated with Density based semi empirical Model (DBSE) equations beyond the regions and scope of this measured empirical data. With this as the goal, the predictive modeling program described here has been written and focused to meet this expectation. Further, the complete work (pipeline) presented here, is also designed for visualization and for explicating the phase behavior of existing (and newer) models and for estimating the boundedness of the estimated parameter space. This holistic approach in this pipeline is useful for designing newer, efficient (accurate and precise) equations heuristically. Conveniently, As previously mentioned, a one-time-run-all code has been provided for implementing state-of-the-art machine learning algorithms for predictive modeling of DBSE model equation associated data. When correctly deployed, this work could potentially reach the helm of this growing body of research from this three-pronged computational modeling approach. To summarize, the programs are stand alone, simple, unique, computationally economic and are also easy to implement. The objectives and Software being postured here in this article are listed below,

1. A MATLAB program for estimating and comparatively analysing, parameters of extant/newly developed density based semi empirical model equations of supercritical fluid extraction process comprising of variables [ln(y), T, P, D] using ordinary least squares parameter estimation method.
2. A MATLAB program for visualizing parameter profiles and Phase behaviors of DBSE model equations using 3D surface plots.

3. A Python based Jupyter Notebook for implementing supervised machine learning algorithms (Multilayer Perceptron, K nearest neighbours and Support vector machines) based on experimental data involving the variables (Temperature (T), Pressure (P), Density (D) and Solute solubility Mole fraction (y)).
4. Provide concluding remarks about the program scripts, its usage and availability.

Experimental

Description of Data: Input Matrices and Parameter Description

The MATLAB (Matlab 1984) and Python program scripts presented in this work requires two input matrices. First, Consider, the Input data as a matrix where in, $Data \in R^n$ and $n \in Z$, then,

$$Data_{i,4} = \begin{bmatrix} T_{1,1} & P_{1,2} & D_{1,3} & y_{1,4} \\ \vdots & : & : & \vdots \\ T_{i,1} & P_{i,2} & D_{i,3} & y_{i,4} \end{bmatrix} \quad (1)$$

Where T is temperature in Kelvin, P is pressure in Mpa, D is density in Kg/m3 and y is solubility mole fraction of the solute in the reaction mix. The index ‘i’, runs over the entire column length of a single feature. This is the first input data matrix required and is parsed by the scripts via the Input_Data.xlsx file.

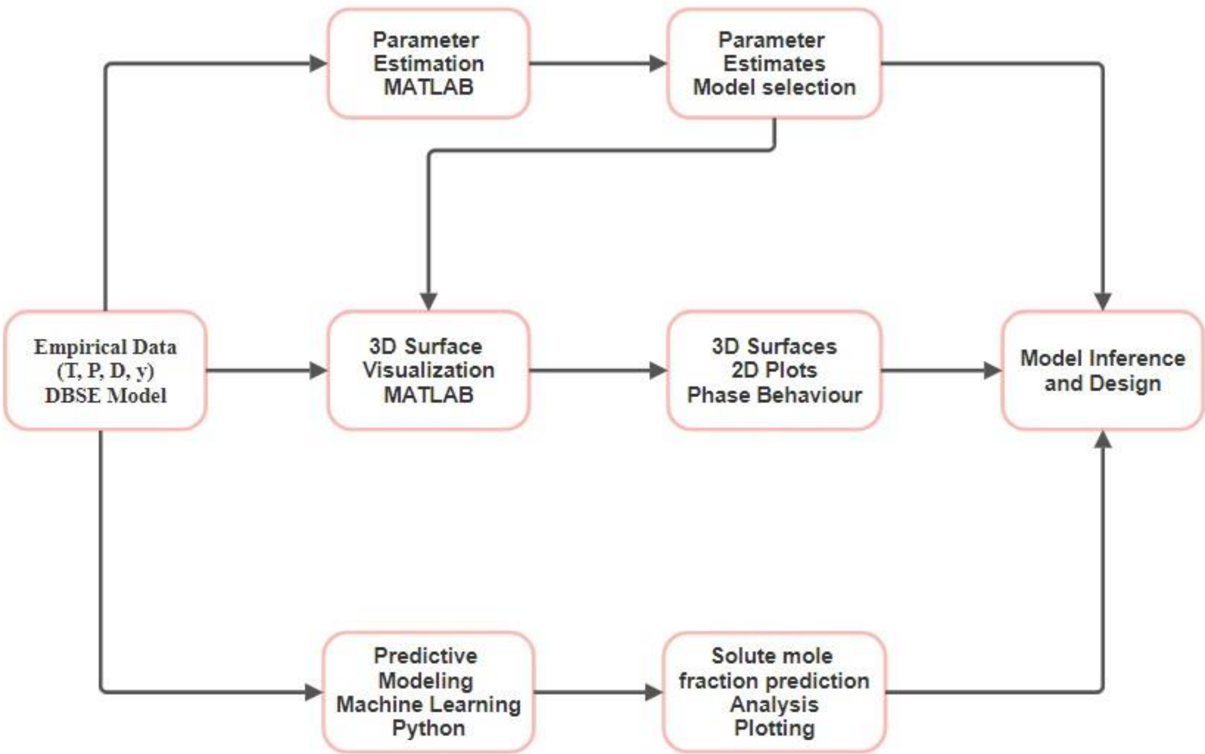


Fig. 1 Flow chart illustrating a single iteration by the parameter estimation, 3D visualization and Predictive modeling program scripts

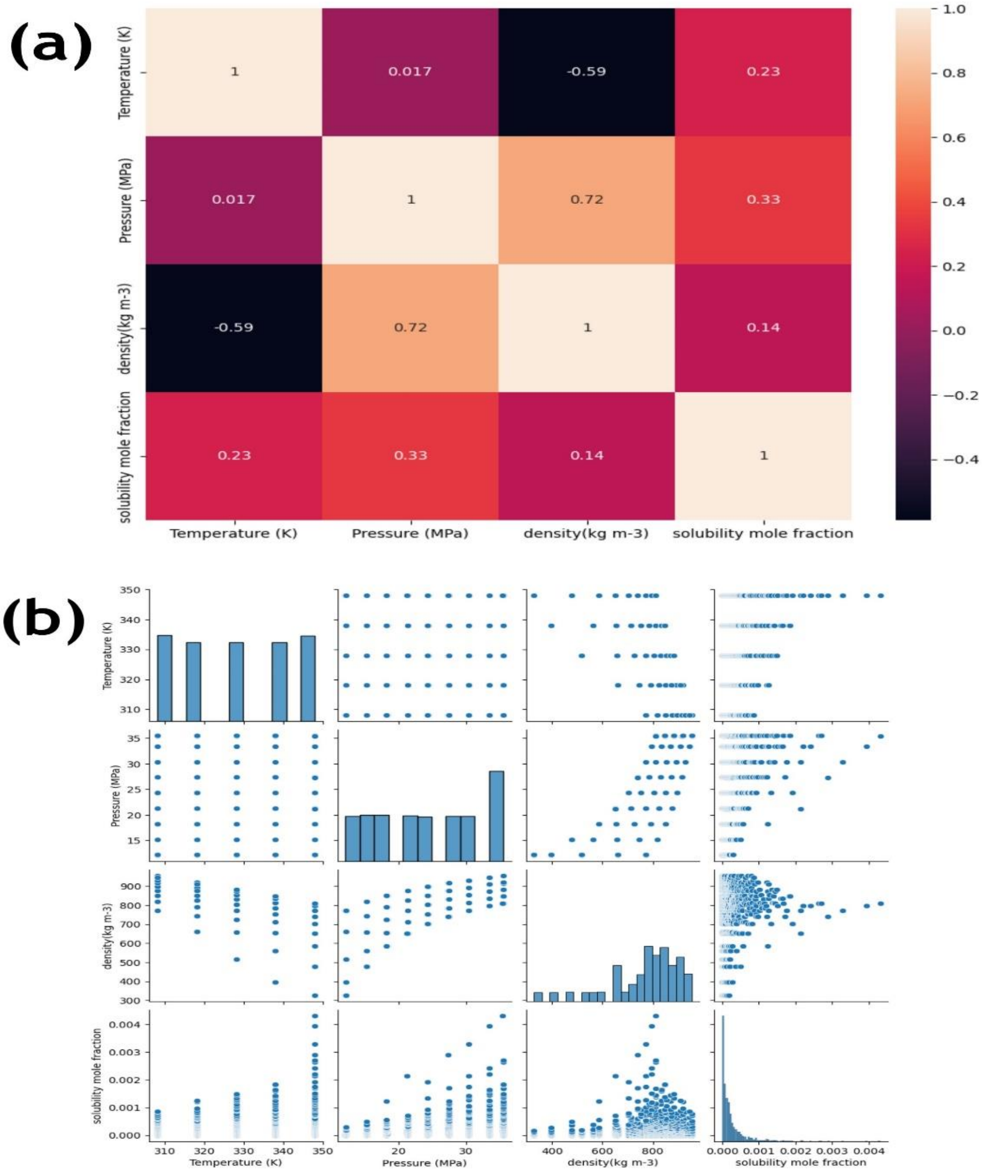


Fig. 2 (a) Heat Map plot of correlation values of input parameters (Temperature, Pressure, Density and Solute Mole fraction). (b) Parameter pair plot of data points including all combinations of input parameters for illustrating patterns present among variable pairs

The second matrix required by the program scripts is comprised of the terms of the input density based semi empirical equations. For illustration, consider a simple four parameter linear model equation and its basic generalization,

$$\ln(y) = A + B[T] + C[P] + D[\rho] \equiv Y = p_1[Term1] + p_2[Term2] + p_3[Term3] + p_4[Term4] \quad (2)$$

Where, A, B, C, D corresponds to p1, p2, p3, p4 and are the parameter coefficients or estimands of the DBSE model equation given above (however, users can input any number of terms). Let these parameter coefficients be grouped into vector ‘P’. Let the terms of the model [term1, term2, term3, term4] be grouped into a vector named as ‘Terms’. For the estimation of the model coefficients [P] and for obtaining parameter estimates $\hat{p}$, the terms of the sampled DBSE model equations are input into respective cells of rows particular to each model equation in a separate .xlsx file (Models_Equations.xlsx). These are the two input matrices (in .xlsx format) required by the MATLAB based parameter estimation script and the visualization script. A toy input data sample containing 1000 experiments along with a sample of ten randomly selected, semiempirical equations have been used for producing the output presented in this article. The modification path traversed by the data in a single iteration by the program scripts is illustrated (Fig. 1). Also, the Input data is analyzed using the Jupyter Notebook and the outputs (Correlation heat map and parameter pair plot) are depicted (Fig. 2 a**,** b). Refer to the user guide (given in the repository) for information on using these program scripts for custom data and model equations (existing/newly proposed). The user guide also provides information regarding the preselection of the base model along with the descriptions of the randomly sampled model equations present in the unmodified file (Models_Equations.xlsx) for ease of usage.

Parameter Estimation: Ordinary Least Squares Method

Estimation of parameter coefficients represented in the vector P is performed using the method of Ordinary Least Squares Parameter Estimation (Dismuke and C R Lindrooth 2006) by the MATLAB program script (DBSE_OLS_Estimation.m). A concise development of the implemented algorithm is presented. Consider a representation of a DBSE model equation in the form of the classical linear regression model,

$$Y_{i,1} = [Terms\,]_{i,k}[P] + \varepsilon_{i,1} \quad (3)$$

Let the assumptions, about the error in the models be, errors are additive, uncorrelated, has zero mean and has constant variance.

Also,

$$E(\varepsilon\varepsilon^T) = \sigma^2 I_i \quad (4)$$

Where ε is the residual vector and σ^2 is the variance of the residual. Further, let the data substituted, matrix of terms ‘Terms matrix’ be represented for brevity as X and let Y be the vector of natural logarithm of solubility mole fraction values. Then the ordinary least squares estimator $\hat{p}$ is given by,

$$\hat{p} = [X^T X]^{-1} X^T Y \quad (5)$$

The vector of residuals ε is given by,

$$\varepsilon = Y - X\hat{p} \quad (6)$$

The confidence intervals (bounds) of the estimates are computed at 95% confidence level. Further, model selection is iteratively performed using an F-Statistic score (Belitser et al. 2011) for each model equation relative to a preselected base model (This is input in the first row of the Models_Equations.xlsx file). Let the residual sum of squares for the DBSE model of a particular iteration and the same for base model be,

$$R_{ols}^{model} = \varepsilon_{ols}^{T}\varepsilon \; R_{ols}^{base} = \varepsilon_{base}^{T}\varepsilon \quad (7)$$

Then the equation for an F-score metric-based model selection is,

$$\frac{\left(R_{ols}^{base} - R_{ols}^{model}\right) / \left(n_{p,0} - n_{p,base}\right)}{R_{ols}^{model} / \left(n - n_{p,0}\right)} > F_{\left(n_{p,0} - n_{p,base}\right),\left(n - n_{p,0}\right)}^{0.05} \quad (8)$$

Where $n_{p,0}$ is the number of parameters in the current iteration and n is the number of data points (experiments) in the parsed input data and $n_{p,base}$ is the number of parameters in the base model. In the data driven paradigm where modeling is focused on fitting a specific sample of empirical data, this automated selection procedure is beneficial for decimating lower quality equations and for identifying the most contextually appropriate one. Further, error metrics namely, mean squared error (MSE), Root Mean Squared Error (RMSE), Mean Absolute error (MAE) and Percentage Absolute Average Relative Deviation (% AARD) were computed between experimental and predicted solubility using the expressions,

$$Mean\ Squared\ Error = \frac{1}{n}\sum_{i=1}^{n}\left(\ln{(y)_i^{pred}} - \ln{(y)_i^{exp}}\right)^2 \quad (9)$$

$$Root\ Mean\ Squared\ Error = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\ln{(y)_i^{pred}} - \ln{(y)_i^{exp}}\right)^2} \quad (10)$$

$$Mean\ Absolute\ Error = \frac{1}{n}\sum_{i=1}^{n}\left|\left(\ln{(y)_i^{pred}} - \ln{(y)_i^{exp}}\right)\right| \quad (11)$$

$$\%AARD = \frac{100}{n}\sum_{i=1}^{n}\frac{\left|\ln{(y)_i^{pred}} - \ln{(y)_i^{exp}}\right|}{\ln{(y)_i^{exp}}} \quad (12)$$

Error metrics have been computed using natural logarithm of solubility mole fraction values for predictions after parameter estimation and actual solubility mole fraction values have been used for predictions from predictive modeling.

Visualization of Phase Behaviour Projected by DBSE model Equations:

Visualization of Phase behavior using three dimensional surfaces of the input DBSE model equation is implemented using MATLAB program. The MATLAB script

(DBSE_3D_Viewer.m), requires, model equations and empirical data (Input_Data.xlsx and Models_Equations.xlsx) along with the estimates (Parameter_Predictions_Results.xlsx) and iteratively plots three dimensional surfaces of the model equations using finitely spaced grid points of the parameters present in the particular DBSE model equation in the iteration.

Three surfaces are plotted by this script namely, Pressure-Temperature-Solute mole fraction, Density-Pressure-Solute mole fraction and, Density-Temperature-Solute mole fraction. Standard, inbuilt commands from MATLAB are used for plotting the surfaces for all of the input DBSE model equations. The output images are also in the standard interactive MATLAB plot window (environment) which allows for altering values of axes to obtain surfaces (Rovenski 2010). Notedly, empirical data is used by this MATLAB program only for finalizing extreme values of the grid points used for plotting these surfaces. Therefore, the surfaces plotted by this script illustrate phase behavior and vapor liquid equilibrium data projected by the specific DBSE model equation and these surfaces are not influenced by the pattern prevalent in the input empirical data. Finally, this MATLAB program exports all three surfaces plotted for a DBSE model equation as subplots in a single image (.jpg) format.

Prediction of Solute Solubility: Machine Learning Algorithms

Three Supervised Machine learning algorithms have been implemented using the Python module, Sklearn (Pedregosa et al. 2011) in a single Jupyter notebook (DBSE_Predictive_Modeling.ipynb) (Menke 2020). This Notebook, using input empirical data, in a single run, implements the Multilayer perceptron, K-nearest Neighbours regression and Support Vector regression algorithms before performing detailed and comparative analysis on the predictions and results. Standard metrics are used for performing validation and analysis of results. Numpy (Oliphant 2006), Openpyxl, Pandas (W McKinney 2011), Matplotlib (Hunter 2007) are among the python packages used for implementing these algorithms. The script requires empirical data (experiments in rows complete with Pressure, Temperature, Density and the resultant, solute mole fraction) characteristic to density based semi empirical model equations. Also, the input parameter space is not exhaustive and can incorporate additional parameters based on user preference. Descriptions of the implemented algorithms and their tuneable hyperparameters are provided in the subsequent paragraphs.

Multilayer Perceptron Regression [MLP]

Multilayer Perceptron [MLP] is a fully connected class of feed forward artificial neural networks classified as a supervised machine learning algorithm. This framework consists of updatable, weight assigned nodes called neurons that are sorted into three types of fully connected layers namely, input layer, hidden layer(s) and an output layer. During the training of a single instance (experiment), parameter (feature) information is fed into the input layer which is then transmitted to the next hidden layer(s) where activation function(s) modify this information for final modification in the output layer. The output layer, using an activation function, modifies the received information and provides data output. This output is the prediction value of the algorithm. Information modification during training (learning) results in the updation of the initialized weights (associated with neurons and connections) from the previous learning iteration (Murtagh 1991). In this MLP model, for obtaining accurate and precise output (solute solubility mole fraction), hyperparameter search space for size of hidden layer, neurons, activation functions, learning rate, data split ratio, solver, alpha value etc can be easily optimized in the notebook based on user preference and data. Theoretical explanation and development of the MLP algorithm can be obtained in literature elsewhere (Schilling et al. 2015). The results and analysis from this program code are exported to an excel notebook (Ml_Results.xlsx).

K-Nearest Neighbours Regression [KNN]

K- Nearest Neighbours algorithm is a non-parametric, supervised machine learning algorithm. For regression problems, the algorithm learns to predict the target class value based on the k closest training examples (instances or experiments) in the input data. The model during learning (training), performs search in the data pattern space for the closest number of training instances. The results from this search which are the closest ‘k’ number of training instances (neighbours), are averaged to obtain the prediction value (solute solubility mole fraction) during testing (Kramer 2013). The adjustable/tuneable hyperparameters for this algorithm is the ‘k’ value (sampling metric) and the distance (closeness) measurement metric (Cunningham and Delany 2022). Here in this jupyter notebook, Euclidean distances are calculated to measure closeness for the preassigned k value which is used to obtain a detailed, comparative, analysis of the prediction which are exported (Ml_Results.xlsx).

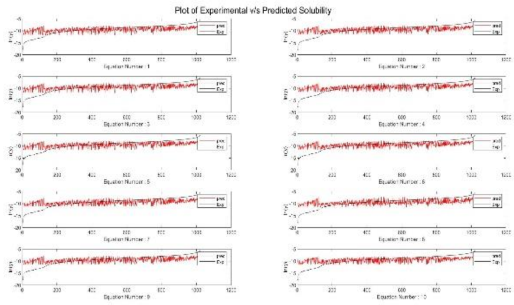
Support Vector Regression [SVR]

The support vector regression algorithm is a class of support vector machine algorithm and is also a supervised machine learning algorithm. In fewer sentences, support vector regression algorithm, using a kernel function, tries to map the input parameter variable data to a feature space (usually of higher dimension) and with the aim of minimizing prediction error, tries to find a hyperplane in this feature (parameter) space that maximizes the distance margin between this plane and the closest data points. Theoretical development of the SVR technique and the mechanism behind its prediction capabilities can be obtained in detail here (Smola and Schölkopf 2004). The tuneable hyperparameters here are the kernel function, gamma value and the test-train data split ratio. Scaling of the parameter data has not been implemented in this jupyter notebook for SVR because the pattern present in the parameter data are highly relevant for the prediction of the solute solubility mole fraction (Tsirikoglou et al. 2017). The jupyter notebook, after implementing support vector regression, separately provides results which is also exported (Ml_Results.xlsx).

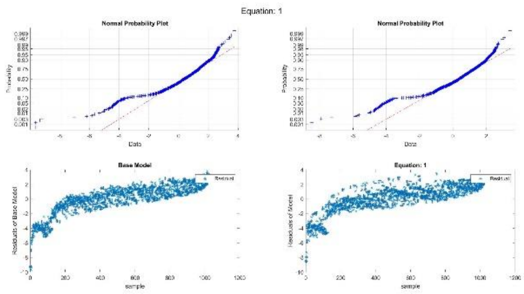
Results and discussion

Parameter Estimation: Ordinary Least Squares Method

(a)



(b)



(c)

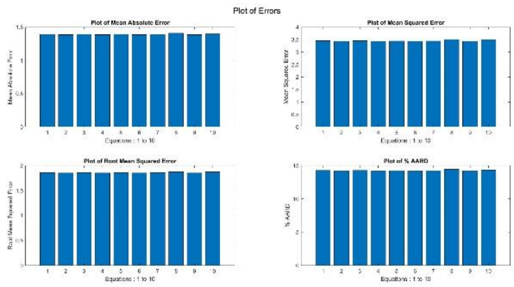


Fig. 3 Standard output for the model equation(s) being iterated from the MATLAB based parameter estimation script. (a) Plot of Experimental (black) v/s Predicted (red) values of the natural logarithm values of solute solubility molefraction. (b) Plot containing normality plots and residual plots for base model of choice and the model being iterated. (c) Bar plots pertaining to error metrics for all input equations

As previously derived, A customized Ordinary least squares estimation method has been implemented to obtain parameter estimates of model equation constants along with confidence intervals in a 'one model equation at a time' iterative rule fashion. This provides the users, with parameter (model coefficients) estimates from a standardized and commonly used method for all model equations in the batch sample (input using an .xlsx file). Confidence intervals (upper and lower bounds) are estimated for each parameter estimate at 95 percent confidence. Conveniently, the results are saved into an excel workbook. The pictorial output from the program script are presented in (Fig. 3 a – c). Natural logarithm values of solute solubility mole fractions are plotted against number of experiments for both empirical data and predictions made using the estimated parameters (model constants) and state variables (Pressure, temperature and Density) associated with the model equations. Normality plots and residuals of the base model and the model equation (being iteratively estimated) are also charted for ascertaining the nature of the data as it is a necessary condition. The normality and residual plots are showcased (Fig. 3b). Normality plots reaffirm and ascertain the considered assumptions about the residuals while estimating parameter coefficients (Model constants). This step makes sure the estimates are contingent with the assumptions made regarding the data and by extension, also the residuals. In the Fig. 3 b above, the data appear to lie on the line of reference demonstrating the degree of normality present in the sample data. Unfortunately, the large amount of data (from the toy data sample) in the residuals plot indicate a pattern and masks the randomly distributed points in the region of interest. This region of interest corresponds to the operating conditions where solute solubility is supposedly maximum (window of maximum solubility). However, this also will change when different empirical data is used. The results from model selection (outperforming model equation) based on the previously mentioned F-Scores are also stored in the excel workbook associated with the predictions from model equation and its estimates. Scores computed from F Distribution, provide clear, statistical comparison between the model equation being iteratively estimated and the base model equation of choice (Input in the first row in the Models_Equations.xlsx file).

Additionally, users can easily make excellent inference based on published literature regarding the estimates and selection output produced by this program (Garlapati and Madras 2010; Reddy and Madras 2011; Bian et al. 2016; Alwi and Garlapati 2021b). The pictorial illustration indicates the plotting constraints (maximum number of subplots in the image output) associated with the presented code and users are encouraged to consider this factor while sampling model equations. Plotting natural logarithm values of the predicted data against actual solute solubility mole fraction values of the predicted data (from model equations), provides clear distinction and higher resolution of model fit and deviation from empirical data. Errors and residuals are also calculated using natural logarithm values for this important reason. In reality, based on the toy sample empirical data, the error metrics and residuals appear to be significantly (desirably) low when actual solubility values are used as opposed to their natural logarithm values. Mean squared error (MSE), Root Mean Squared Error (RMSE), Mean Absolute error (MAE) and Percentage Absolute Average Relative Deviation (% AARD) values are computed using Eq. (9)-(12), plotted and presented in the form of bar graphs in a single image format (Fig. 3 c). Errors of all model equations appear to only slightly differ indicating superior quality of the sampled toy data. However, as previously mentioned, this too will differ for other empirical data. Due to constraints for assessing and visualizing higher numbers of equations, sampling (ten to fifteen equations) and selection of model equations must be of higher quality. However, the provided code for batch estimation (DBSE_OLS_Estimation_Batch.m), has full capability to estimate one hundred DBSE model equations in a single implementation run. In summary, this program script provides parameter estimates of model(s) coefficients along with their confidence regions (intervals). Further, Model selection and identification routine also aids in comparative assessment and selection of the best performing model equation all of which are then exported to popular file formats.

Visualization of Phase Behavior of DBSE Model Equations:

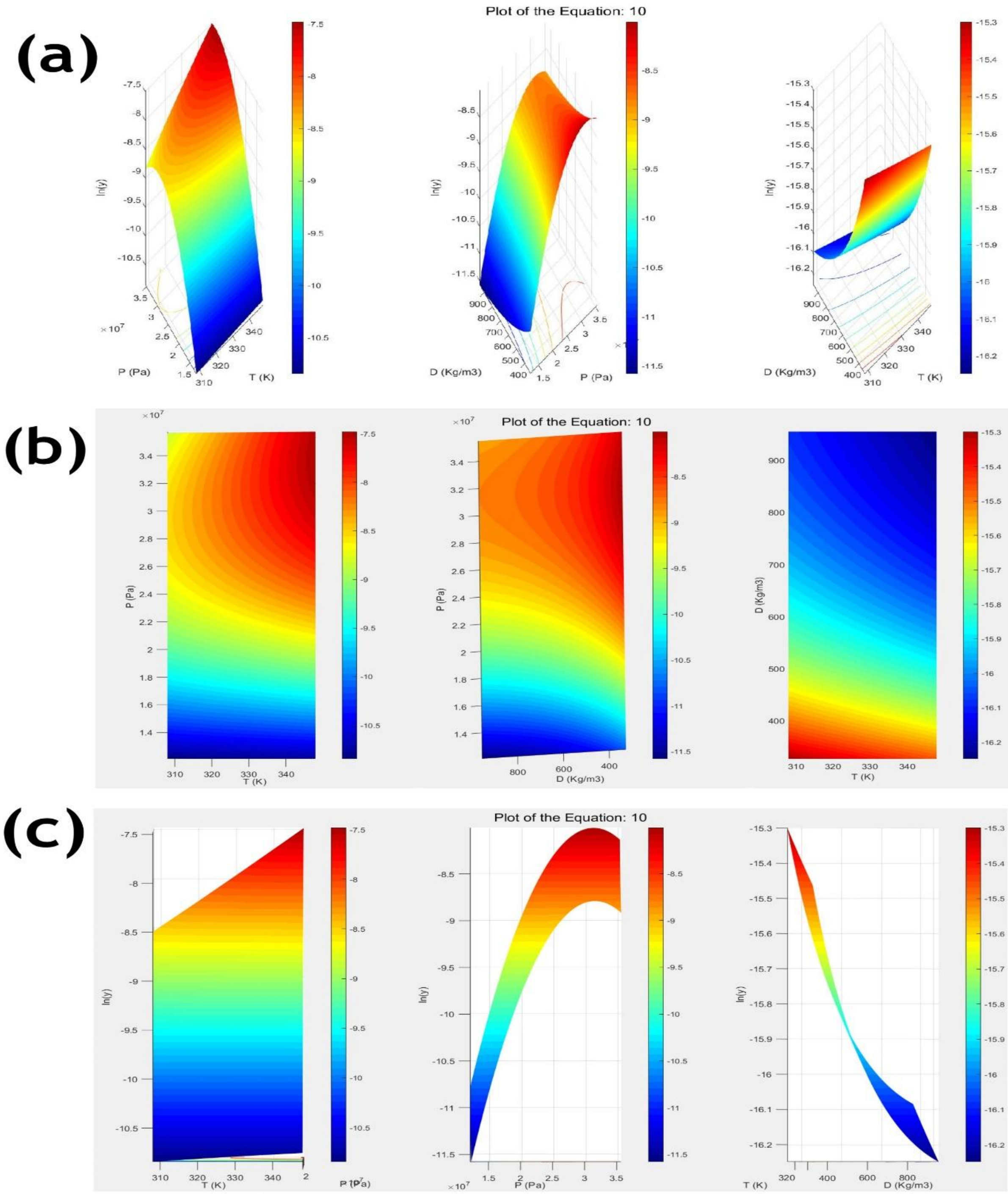


Fig. 4 Three dimensional surfaces of ln(y)-P-T, ln(y)-D-P, ln(y)-D-T. (a) This plot is the only standard output produced by the MATLAB based visualization script. (b) Two dimensional, color coded contour plot of P-T, P-D, D-T obtained from the same MATLAB interactive plot window. The projections for these plots are visible on the respective 3D surface (a). (c) Two dimensional, color coded contour plot of ln(y)-T, ln(y)-D, ln(y)-D obtained from the MATLAB interactive plot window

The three-dimensional surfaces of the P-T-D state variables and the natural logarithm values of solute solubility mole fraction obtained from the MATLAB script for visualization is illustrated in Fig. 4 a – c**.** The interactive nature of the MATLAB surface plot window and the ease with which axes values of the surface can be altered makes the obtained pictorial output very valuable for evaluating the phase equilibria characteristic to the specific DBSE model equation. Fig. 4 a shows a grab of the three surfaces [P-T-ln(y), P-D-ln(y), T-D-ln(y)] arranged as subplots from a single interactive (image) window output. Grabs of two-dimensional plots (Fig. 4 b – c) can be obtained from these surfaces by independently altering the axes values of the surfaces in the interactive MATLAB plot window. The surfaces are primarily color coded to indicate the gradient in solute solubility. Projections of these surfaces manifest as grid lines (phase curves of ln(y)) on the axes planes. These plots provide insight regarding the major and minute differences in the projected phase behavior put forth by the model equations. Conveniently, even small or minute variations in a combinatorial pool of model equation designs (derived from a single parent equation) manifests acutely in the shape and color gradient of the corresponding surface plots (Goos et al. 2011; Yamini and Moradi 2011; Cockrell et al. 2021). Further, literature (Schneider 1978) can be referred to make accurate inferences regarding model specific phase behavior from these surfaces and projections. However, a probable/possible approach for gaining satisfactory information from these surfaces (3D), its derivative plots and plane projections (2D) is provided below.

Consider a set of model coefficient parameter estimates (from a DBSE model equation) of a (sufficiently) well modelled super/sub critical fluid extraction process (for example, coffee or tea decaffeination) pertaining to a ternary system of CO2/H2O solvent, Co-solvent (Ethanol or methanol) and solute (This is subject to availability and procurement by the user and is not provided here in this article). Let this set of obtained estimates be then used to plot the 3D surfaces and derivative plots. Naturally, due to the process being sufficiently well modelled (as previously assumed), knowledge regarding the Phase diagrams, vapor-liquid equilibrium behavior, maximum solubility window and equilibrium points and planes is readily available, importantly reliable and trustworthy for these estimates, plots and the associated empirical data. Let this information (again, not provided here with this article) be the ground truth and basis for performing further comparative analysis using the MATLAB based plotting and visualization script presented here in this article. Then the surfaces and 2D projections obtained by implementing this script for the same empirical data (and the model coefficient estimates) for a batch of DBSE model equations (existing/newly developed) can now be used to evaluate and glean information regarding important attributes like the upper and lower critical end points, planes and edges associated with the triple point. Further, vapor pressure curves and the data characteristic to the components (pure and mixture) in the ternary system can be identified and compared to this ground truth.

Generally, the qualitative and quantitative data regarding the latency, miscibility, compression, crystallizability of the components in the reaction mix can be obtained from these surfaces. Further, the identification of solid-liquid-gas lines describing boundaries of latency (or miscibility) projected by the specific DBSE model equation can also be compared to this truth (if available) and the error values quantify deviation and subtle / major differences. Similarly, values of slope differentials (dP/dT, dT/dD and dP/dD) are easily computed from the surfaces for these equations. These slope values are important for identifying upper and lower crossover pressures bordering the retrograde solubility region in the phase diagrams for explaining retrograde solubility interference (Foster et al. 1991; Esmaeilzadeh and Goodarznia 2005; Kalikin et al. 2021). This approach could be very important and beneficial for comparatively evaluating newly designed DBSE model equations regarding the maximum solubility window and the above-mentioned attributes. This comparative evaluation can then be used for redesigning customized, newer and efficient model equation alternatives. Note that the maximum solubility window depicted in Figure 4(b) is predicted and shown to lie somewhere around the red regions (probably between 320K-340K and 30-32 MPa) by the tenth model equation (from the same randomly mined sample of ten input equations). As pictorially showcased, this too will differ for different equations for the same data. In summary, the plots provide satisfactory, quantitative and qualitative knowledge regarding the phase behavior and equilibria characteristic to the equations being studied, using this MATLAB based plotting and visualization script.

Prediction of Solute Solubility: Machine Learning Algorithms

Multilayer Perceptron regression (MLP), K-Nearest Neighbours regression (KNN) and Support Vector Regression (SVR) algorithms have been implemented using 'sklearn' package in python in a single jupyter notebook. A toy data sample of 1000 randomly mined experiments are used to illustrate the working of this jupyter notebook. The input parameters present in the toy data sample are Temperature, Pressure and Density. The target / output / dependent variable is the Solute solubility mole fraction. Additional parameters can be easily incorporated into the data sample by simply concatenating them as columns after the Density data column in the input Excel workbook (Input_Data.xlsx). The notebook initially provides the description of the data by using basic statistical metrics (count, mean, standard deviation, minimum and maximum value), Correlation values between the parameters and output, Heat map of correlation values and a parameter pair plot for comparing all parameter pairs (combinations) on a chart. These charts are depicted in (Fig. 2 a – b). The results (graphs, errors and plots) and discussion pertaining to each algorithm is provided in the subsequent paragraphs.

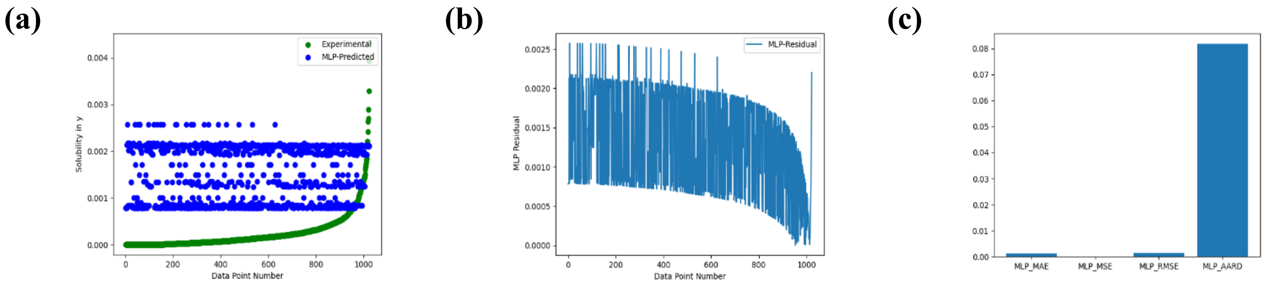


Fig. 5 Standard output from the jupyter notebook about the predictions and analysis of the Multilayer perceptron algorithm (a) Scatter Plot of Experimental (green) v/s Predicted (blue) values of solute solubility molefraction from the Multilayer perceptron algorithm. (b) Plot of residual values from the Multilayer perceptron algorithm. (c) Bar plot of error metrics of the predictions from the Multilayer perceptron algorithm

Multilayer Perceptron regression (MLP) is the first algorithm implemented in this Jupyter notebook. Data scaling (preprocessing) is performed using the ‘MinMaxScaler’ routine before further transformation of the data. The data is then split (preprocessing) using the test-train-split routine. The results and the output obtained are illustrated in Fig. 5 a – c. Regression model is built using the standard ‘MLPregressor’ routine. Hyperparameter optimization / tuning is performed by using the ‘GridSearchCV’ routine for the MLP algorithm. As explained, users have to define the space for grid search for the hyperparameters (Number of hidden layers, activation functions, solvers, learning rate) in the beginning of the notebook for hyperparameter optimization. Further, 5-fold cross validation is performed based on negated values of root mean square error as the model scoring metric. The program performs tuning and the hyperparameters of the best model are then used to refit and obtain the prediction output (Schilling et al. 2015). Error metrics for this algorithm are (output) plotted (Fig. 5 c) separately for brevity.

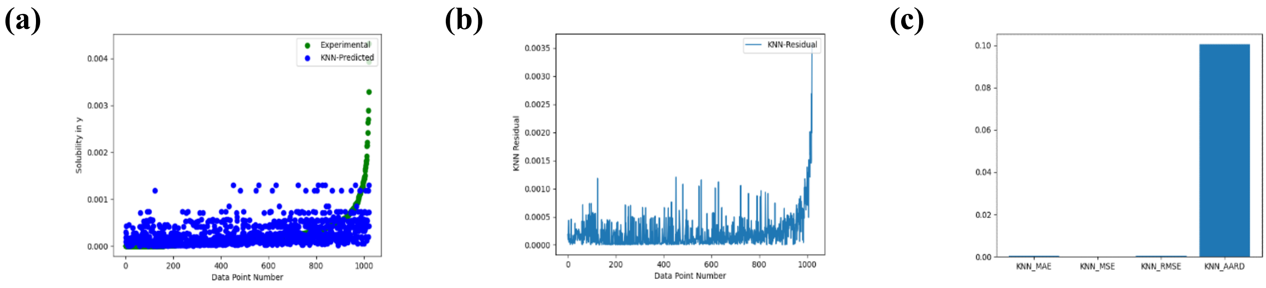


Fig. 6 Standard output from the jupyter notebook about the predictions and analysis of the K-Nearest Neighbours algorithm. (a) Scatter Plot of Experimental (green) v/s Predicted (blue) values of solute solubility molefraction from the K- Nearest Neighbours algorithm. (b) Plot of residual values from the K- Nearest Neighbours algorithm. (c) Bar plot of error metrics from the K- Nearest Neighbours algorithm

K-Nearest Neighbours regression (KNN) is implemented next (after MLP) in the notebook. As discussed, Data scaling was deemed unnecessary and has not been performed. However, test train split is performed using the same routine as MLP. Further, The Hyperparameter K is set to a random value of 3 for the toy data sample and can easily be changed / tuned based on user data and preference at the beginning of the notebook. Error metrics for the KNN algorithm is plotted (Fig. 6 c) separately. Further insight regarding the model can be obtained from data, hyperparameter optimization and previous literature (Soleimani Lashkenari and KhazaiePoul 2017).

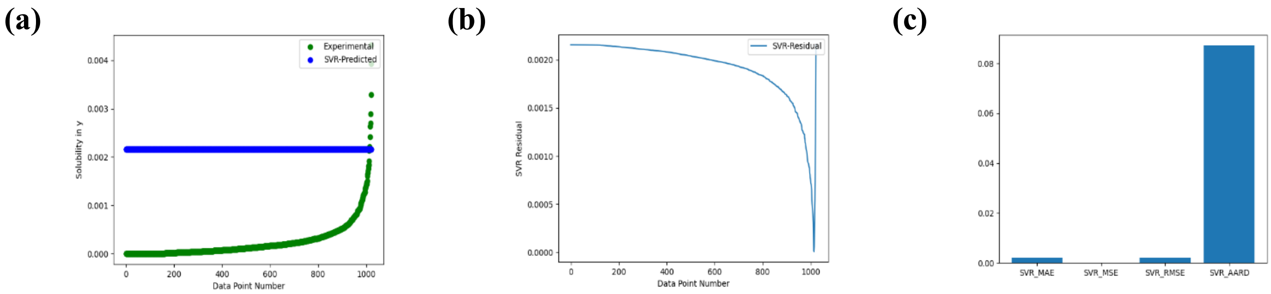


Fig. 7 Standard output from the jupyter notebook about the predictions and analysis of the Support Vector Regression algorithm. (a) Scatter Plot of Experimental (green) v/s Predicted (blue) values of solute solubility molefraction from the Support Vector Regression algorithm. (b) Plot of residual values from the Support Vector Regression algorithm. (c) Bar plot of error metrics from the Support Vector Regression algorithm

Support Vector Machine Regression (SVR) algorithm is implemented at last in the notebook. Like before, data scaling is not performed so as to preserve pattern in the input parameter space. Data has been split for model training using the test – train split routine like before and can be easily adjusted by the user. The choice of Kernel function hyperparameter is also tuned using 'GridSearchCV' and users can modify the grid search space for this at the beginning of the notebook. Five-fold cross validation is performed based on the negated root mean squared error scoring metric and the kernel function associated with the best scoring model is then used to refit and obtain the predictions (Tsirikoglou et al. 2017). Error metrics for SVR, like before, is also plotted (Fig. 7 c) separately.

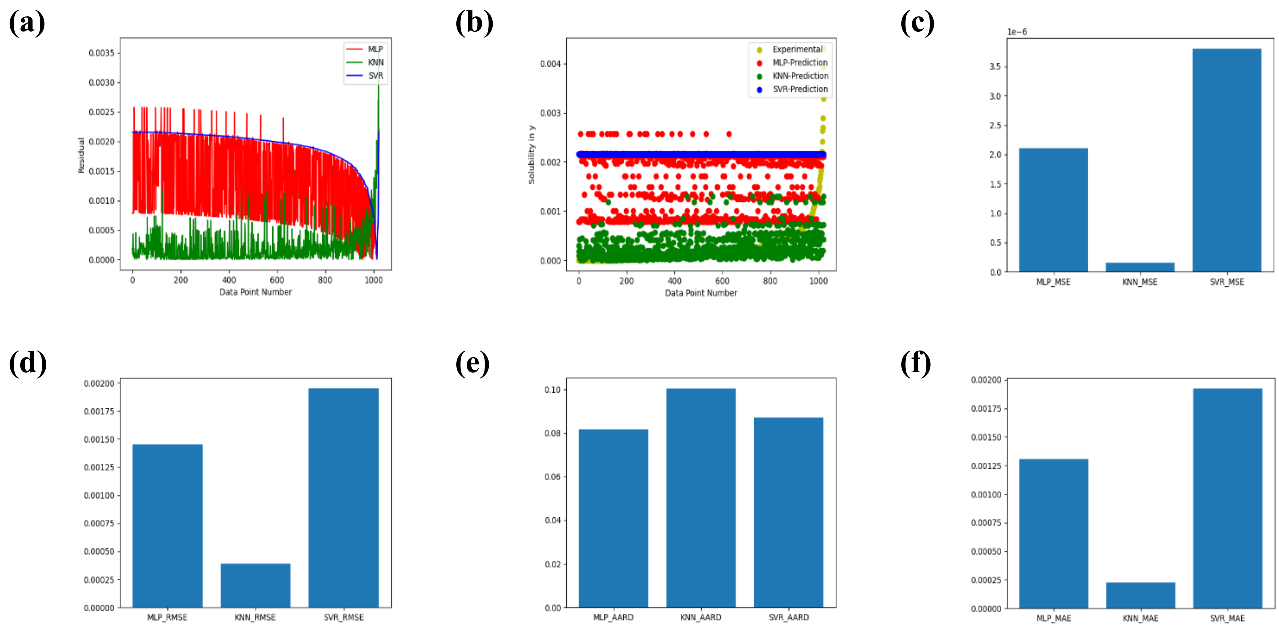


Fig. 8 Standard output from the jupyter notebook about the predictions and comparative analysis of all three algorithms from each complete program run. (a) Combined plot of residual values of all three machine learning algorithms (MLP, KNN, SVR) for comparison (b) Plot of Prediction values of KNN and SVR algorithms with experimental solubility values. (c) Bar plot of mean squared error values of all three machine learning algorithms (MLP, KNN, SVR) for comparison. (d) Bar plot of root mean squared error values of all three machine learning algorithms (MLP, KNN, SVR) for comparison. (e) Bar plot of percent absolute average relative deviation values of all three machine learning algorithms (MLP, KNN, SVR) for comparison. (f) Bar plot of mean absolute error values of all three machine learning algorithms (MLP, KNN, SVR) for comparison.

Model fitting and prediction of all three algorithms (MLP, KNN and SVR) for the sample data yielded results and the computed errors (MSE, RMSE, MAE and %AARD) are plotted separately on bar plots (Fig. 8 c**,** d**,** e, f). The predictions v/s empirical data graph is also plotted and exported by the notebook (Fig. 8 b). Likewise, residuals are also plotted for all three algorithms (Fig. 8 a). Hyperparameter tuning for all three algorithms is implemented and other intricate nuances pertaining to the predictions can be easily made by the users based on the best performing algorithm and the input data (Feurer 2019). The parameter space, as previously explained, can (only) be increased by the user to explore and incorporate additional parameters like Melting point, Boiling Point, Total polar surface area, Critical Temperature, Critical Pressure, Molecular Weight of solute, percentage of co-solvent used, type of cosolvent (by scoring) etc. Therefore, Detailed explanation regarding the obtained numerical output is unnecessary here since a toy data sample with the standard (Temperature, Pressure and Density) parameters have been used. However, users can glean and infer information from their custom empirical data using these plots and tables which are produced for each algorithm by this jupyter notebook. The notebook has been written to include the best of the plot commands and features (errors, functions, tables etc) from standard python libraries for ease of use and assessment. The numerical data predictions and analysis are exported to an excel workbook. Importantly, users are cautioned against the usage of this notebook for actual experimental purposes as it can be dangerous when used directly in a laboratory setting without proper consultation. The provided notebook is an efficient tool for data analysis and is very useful for theoretical research, modeling (fitting), understanding and comparison. Overall, The Jupyter notebook provides the users, with a state-of-the-art predictive modeling and analysis tool using standard Machine learning algorithms for obtaining prediction values of solute solubility mole fraction from input parameter data.

Conclusions

This work showcases program scripts and their workflow (pipeline) as a comprehensive, state of the art parameter estimation and predictive modeling tool for evaluating density based semi empirical models (equations) and its associated data. Parameter estimation has been implemented in a MATLAB based script using a customized version of the popular Ordinary least squares estimation method. The programs are stand alone in that they fully function even when the parameter estimates for input equations are externally sourced. Further in this work, Visualization of phase behaviours projected by preselected (sampled) model equations using a MATLAB based script has been described. This visualization script produces three-dimensional surface plots in interactive MATLAB windows based on the parameter estimates (computed from ordinary least squares estimation). An approach for gleaning theoretical information regarding phase behaviour using the surface plots is provided. Even subtle variations in model equation designs acutely manifests in the shapes and color gradients of the projected surface plots and this makes designing newer, robust, data specific/generalized equations easier. Standard error and scoring metrics have been computed at each appropriate stage in the workflow and presented to users in the form of plot illustrations. Importantly, the maximum solubility window is predicted to lie somewhere around the red regions (probably between 320K-340K and 30-32 MPa) by the tenth model equation (and is predicted for all the remaining input equations). A Python based programming script is also presented for predictive modeling of the empirical data associated with super/sub critical extraction using three Machine learning algorithms. This notebook has been written to accommodate 'n' number of other variables for improving the accuracy of the solute solubility predictions. This allows users with diverse forms of data to easily make predictions, interpretations and reach scientifically sound conclusions about the maximum solubility window. Further, user defined hyperparameter tuning has been implemented for all three algorithms and has not been entirely focused towards fitting the toy data sample (However, the presented error metrics are desirably low). Therefore, Users are strongly advised to use these program scripts for theoretical and academic purposes since these scripts are under continuous development, refinement and modification. The surfaces, plots and tables present in this article are the standard predictions and analysis of outputs from these scripts based on a toy data and model equation(s) sample (mined randomly from literature) and are not regarding any particular density based semi empirical equation or published data. Hence, again, strong caution is advised against their usage directly in an experimental setting without appropriate supervision or reasoning. Importantly, a properly worded guide is provided for using this repository. Future goals include deploying and testing this work on established datasets, similar computational tools, and DBSE model equations. In summary, this work postures a first of its kind, efficient computational tool in the form of program scripts for evaluating/designing Density based semi empirical equations associated with super/sub critical extraction process and data.

Data Availability. The Software programs are available in a GitHub Repository here https://github.com/Srinidhi-hub/DBSE-Evaluator.git . The software programs are also accessible upon request from the author.

Conflict of Interest. The Author declares that there are no conflicts of interests with any entity, institution or individual regarding this study. The author also declares that this study/work did not receive any funding from any source.

Acknowledgements. The Author is grateful to Prof. Dr. Rudiyanto Gunawan (Graduate Academic/Research Advisor), Lab members, Professors, and Colleagues at UB-CBE. The Author is also thankful to Prof. Dr. A. K. Tangirala, Department of Chemical Engineering, Indian Institute of Technology, Madras, TN, IN for hosting freely accessible, insightful code from their book on their website.

Author Biography: Srinidhi received his Master's degree in Chemical Engineering from The State University of New York at Buffalo (CeDiD: 22G6-XJHD-SOI8). Earlier, he received his Bachelor's degree in Biotechnology Engineering from M. S. Ramaiah Institute of Technology. His Academic Interests include Mathematical modeling of Chemical and Biological Processes.

Symbols

T	Temperature	K
P	Pressure	MPa
D	Density	Kg/m^3
R	Residual Sum of Squares	
y	Solute Solubility Mole Fraction	

Greek Letters

ε	*Error*	
σ	*Standard Deviation*	
ρ	*Density*	*Kg/m^3*

References

Alwi RS, Garlapati C (2021a) A new semi empirical model for the solubility of dyestuffs in supercritical carbon dioxide. Chemical Papers 75:2585–2595. https://doi.org/10.1007/s11696-020-01482-x

Alwi RS, Garlapati C (2021b) A new model and estimation of thermodynamic parameters for the solubility of azobenzene and anthraquinone derivatives in supercritical carbon dioxide. Chemical Papers 75:4589–4610. https://doi.org/10.1007/s11696-021-01688-7

Asgarpour Khansary M, Amiri F, Hosseini A, et al (2015) Representing solute solubility in supercritical carbon dioxide: A novel empirical model. Chemical Engineering Research and Design 93:355–365. https://doi.org/10.1016/j.cherd.2014.05.004

Bartle KD, Clifford AA, Jafar SA, Shilstone GF (1991) Solubilities of Solids and Liquids of Low Volatility in Supercritical Carbon Dioxide. J Phys Chem Ref Data 20:713–756. https://doi.org/10.1063/1.555893

Belitser S V., Martens EP, Pestman WR, et al (2011) Measuring balance and model selection in propensity score methods. Pharmacoepidemiol Drug Saf 20:1115–1129. https://doi.org/10.1002/PDS.2188

Bian XQ, Zhang Q, Du ZM, et al (2016) A five-parameter empirical model for correlating the solubility of solid compounds in supercritical carbon dioxide. Fluid Phase Equilib 411:74–80. https://doi.org/10.1016/j.fluid.2015.12.017

Butler KT, Davies DW, Cartwright H, et al (2018) Machine learning for molecular and materials science. Nature 559:547–555. https://doi.org/10.1038/s41586-018-0337-2

Cockrell C, Brazhkin V V., Trachenko K (2021) Transition in the supercritical state of matter: experimental evidence. Phys Rep. https://doi.org/10.1016/j.physrep.2021.10.002

Cunningham P, Delany SJ (2022) k-Nearest Neighbour Classifiers - A Tutorial. ACM Comput Surv 54:1–25. https://doi.org/10.1145/3459665

Dismuke, C R Lindrooth (2006) Ordinary least squares :Methods and Designs for Outcomes Research

Esmaeilzadeh F, Goodarznia I (2005) Supercritical Extraction of Phenanthrene in the Crossover Region. J Chem Eng Data 50:49–51. https://doi.org/10.1021/je049872x

Feurer M (2019) Automated Machine Learning. Springer International Publishing, Cham

Foster NR, Gurdial GS, Yun JSL, et al (1991) Significance of the crossover pressure in solid-supercritical fluid phase equilibria. Ind Eng Chem Res 30:1955–1964. https://doi.org/10.1021/ie00056a044

Garlapati C, Madras G (2010) New empirical expressions to correlate solubilities of solids in supercritical carbon dioxide. Thermochim Acta 500:123–127. https://doi.org/10.1016/j.tca.2009.12.004

Goos E, Riedel U, Zhao L, Blum L (2011) Phase diagrams of CO2 and CO2–N2 gas mixtures and their application in compression processes. Energy Procedia 4:3778–3785. https://doi.org/10.1016/j.egypro.2011.02.312

Hawthorne SB (1990) ANALYTICAL-SCALE SUPERCRITICAL FLUID EXTRACTION. Anal Chem 62:633A-642A. https://doi.org/10.1021/ac00210a722

Herrero M, Mendiola J, … AC-J of C, 2010 undefined (2010) Supercritical fluid extraction: Recent advances and applications. Elsevier 1217:2495–2511. https://doi.org/10.1016/j.chroma.2009.12.019

Huang Z, Shi X, Jiang W (2012) Theoretical models for supercritical fluid extraction. J Chromatogr A 1250:2–26. https://doi.org/10.1016/j.chroma.2012.04.032

Hunter JD (2007) Matplotlib: A 2D Graphics Environment. Comput Sci Eng 9:90–95. https://doi.org/10.1109/MCSE.2007.55

Kalikin NN, Oparin RD, Kolesnikov AL, et al (2021) A crossover of the solid substances solubility in supercritical fluids: What is it in fact? J Mol Liq 334:115997. https://doi.org/10.1016/J.MOLLIQ.2021.115997

Khatib M El, de Jong WA (2020) ML4Chem: A Machine Learning Package for Chemistry and Materials Science. arxiv.org. https://doi.org/https://doi.org/10.48550/arXiv.2003.13388

Knez Ž, Škerget M, KnezHrnčič M (2013) Principles of supercritical fluid extraction and applications in the food, beverage and nutraceutical industries. In: Separation, Extraction and Concentration Processes in the Food, Beverage and Nutraceutical Industries. Elsevier, pp 3–38

Kramer O (2013) K-Nearest Neighbors. pp 13–23

Lakshmi K, Mahaboob B, Rajaiah M, Narayana C (2021) Ordinary least squares estimation of parameters of linear model. Journal of Mathematical and Computational Science 11:2015–2030. https://doi.org/10.28919/jmcs/5454

Matlab (1984) Matlab. The Mathworks.inc

Menke EJ (2020) Series of Jupyter Notebooks Using Python for an Analytical Chemistry Course. J Chem Educ 97:3899–3903. https://doi.org/10.1021/acs.jchemed.9b01131

Murtagh F (1991) Multilayer perceptrons for classification and regression. Neurocomputing 2:183–197. https://doi.org/10.1016/0925-2312(91)90023-5

Oliphant TE (2006) Guide to numpy, 2nd edn. Continuum Press

Pedregosa F, Michel V, Grisel O, et al (2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12:2825–2830. https://doi.org/http://scikit-learn.sourceforge.net/

Rai A, Punase KD, Mohanty B, Bhargava R (2014) Evaluation of models for supercritical fluid extraction. Int J Heat Mass Transf 72:274–287. https://doi.org/10.1016/j.ijheatmasstransfer.2014.01.011

Reddy SN, Madras G (2011) A new semi-empirical model for correlating the solubilities of solids in supercritical carbon dioxide with cosolvents. Fluid Phase Equilib 310:207–212. https://doi.org/10.1016/j.fluid.2011.08.021

Roach L, Rignanese G, Fluids AE-… of S, 2023 undefined (2023) Applications of machine learning in supercritical fluids research. Elsevier 202:896–8446. https://doi.org/10.1016/j.supflu.2023.106051

Rovenski V (2010) Modeling of Curves and Surfaces with MATLAB®. Springer New York, New York, NY

Schilling N, Wistuba M, Drumond L, Schmidt-Thieme L (2015) Hyperparameter optimization with factorized multilayer perceptrons. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9285:87–103. https://doi.org/10.1007/978-3-319-23525-7_6

Schneider GM (1978) Physicochemical Principles of Extraction with Supercritical Gases. Angewandte Chemie International Edition in English 17:716–727. https://doi.org/10.1002/anie.197807161

Schweidtmann AM, Esche E, Fischer A, et al (2021) Machine Learning in Chemical Engineering: A Perspective. Chemie Ingenieur Technik 93:2029–2039. https://doi.org/10.1002/cite.202100083

Selvaratnam B, Koodali RT (2021) Machine learning in experimental materials chemistry. Catal Today 371:77–84. https://doi.org/10.1016/j.cattod.2020.07.074

Smola AJ, Schölkopf B (2004) A tutorial on support vector regression. Stat Comput 14:199–222. https://doi.org/10.1023/B:STCO.0000035301.49549.88

Soleimani Lashkenari M, KhazaiePoul A (2017) Application of KNN and Semi-Empirical Models for Prediction of Polycyclic Aromatic Hydrocarbons Solubility in Supercritical Carbon Dioxide. Polycycl Aromat Compd 37:415–425. https://doi.org/10.1080/10406638.2015.1129976

Tabernero A, del Valle EMM, Galán MÁ (2010) A comparison between semiempirical equations to predict the solubility of pharmaceutical compounds in supercritical carbon dioxide. Journal of Supercritical Fluids 52:161–174. https://doi.org/10.1016/j.supflu.2010.01.009

Tsirikoglou P, Abraham S, Contino F, et al (2017) A hyperparameters selection technique for support vector regression models. Appl Soft Comput 61:139–148. https://doi.org/10.1016/j.asoc.2017.07.017

W McKinney (2011) pandas: a foundational Python library for data analysis and statistics. Python for high performance and scientific computing, . https://doi.org/http://pandas.sf.net

Yamini Y, Moradi M (2011) Measurement and correlation of antifungal drugs solubility in pure supercritical CO2 using semiempirical models. Journal of Chemical Thermodynamics 43:1091–1096. https://doi.org/10.1016/j.jct.2011.02.020