Comparative study of classifier models to assert phase stability in multicomponent mixtures

Lifeng Zhang,a Tanuj Karia,a Gustavo Chaparro,a Kainath Sahebzada,a   
Benoît Chachuat,a Claire S. Adjimana

aDepartment of Chemical Engineering, The Sargent Centre for Process Systems Engineering, Institute for Molecular Science and Engineering, Imperial College London, London, SW7 2AZ, UK

Corresponding Author: c.adjiman@imperial.ac.uk

Abstract

Asserting phase stability entails the global solution of a nonconvex optimisation problem, typically the tangent plane distance minimisation (TPDM). To improve computational tractability, we propose classifier-based surrogate models to replace the TPDM. We seek models that represent several multicomponent mixtures simultaneously, across various component identities, temperatures, and compositions. We investigate both artificial neural networks (ANN) and support vector machines (SVM) and use Matthew’s correlation coefficient (MCC) as performance metric for the corresponding binary classification problems. For SVM models, linear, polynomial, and radial basis function (RBF) kernels are assessed; while for ANNs, the tanh and relu activation functions are investigated. We test the performance of these surrogate models on a set of ternary mixtures that involve ibuprofen and two solvents with fixed or variable temperatures. The results show that ANNs and SVMs can both predict phase stability reliably, with RBF-SVM giving the lowest computational cost.

**Keywords**: Phase stability, Classifiers, Artificial Neural Network, Support Vector Machines, Multicomponent Mixtures.

* 1. Introduction

The ability to guarantee phase stability of a multicomponent mixture is critical in chemical manufacturing. Demixing of a multicomponent mixture is detrimental to the performance and safety of a process or product. Phase stability at a given temperature, pressure and composition is often assessed with thermodynamic models during the design phase by minimising the tangent plane distance (Baker et al., 1982). The nonlinear nature of the underlying thermodynamic model renders the optimisation problem nonconvex, which makes it challenging to obtain a global optimum within reasonable runtime, especially in cases where the problem needs to be solved iteratively. Hence, surrogate models that can be used to determine phase stability have been proposed in the literature, such as artificial neural networks (ANN) for a ternary mixture (Schmitz et al., 2006) and support vector machine (SVM) classifiers for mixtures (Gaganis and Varotsis, 2012). However, most studies focus on constructing surrogate models for a specific mixture, which restricts applications in the solvent design field (Lopez-Ramirez et al., 2023).

In this work, we develop surrogate models that can predict phase stability over a wide range of candidate solvents, with a view to substituting the thermodynamic models embedded in solvent design problems. Specifically, ANNs and SVMs are investigated as the basis for classifier models to determine whether a multicomponent liquid mixture is stable (miscible) under given conditions. Section 2 provides a brief introduction to SVMs and ANNs. The methodology is described in Section 3, while the results are presented and discussed in Section 4.

* 1. Data-driven classifier models
     1. SVM formulation

Given a training dataset with *N* points, where denotes the *i*th feature vector andits corresponding label, the fundamental behind SVMs (Cortes and Vapnik, 1995) is to construct an optimal hyperplane which can separate the positive and negative samplings effectively. The hyperplane can be expressed as follows:

|  |  |
| --- | --- |
| {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi mathvariant=\"bold-italic\">w</mi><mi>T</mi></msup><mi mathvariant=\"bold-italic\">x</mi><mo>+</mo><mi>b</mi><mo>=</mo><mn>0</mn></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (1) |

The weights ***w*** and biases *b* in Eq. (1) are determined by solving:

|  |  |
| --- | --- |
| {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><munder><mi>min</mi><mrow><mi>w</mi><mo>,</mo><mi>b</mi></mrow></munder><mfrac><mn>1</mn><mn>2</mn></mfrac><mo>|</mo><mo>|</mo><mi mathvariant=\"bold-italic\">w</mi><mo>|</mo><msup><mo>|</mo><mn>2</mn></msup><mo>+</mo><mi>C</mi><mstyle displaystyle=\"false\"><munderover><mo>&#x2211;</mo><mrow><mi>i</mi><mo>=</mo><mn>1</mn></mrow><mi>N</mi></munderover></mstyle><msub><mi>&#x3BE;</mi><mi>i</mi></msub><mspace linebreak=\"newline\"/><mi mathvariant=\"normal\">s</mi><mo>.</mo><mi mathvariant=\"normal\">t</mi><mo>.</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><msub><mi>y</mi><mi mathvariant=\"italic\">i</mi></msub><mstyle mathvariant=\"italic\"><mo stretchy=\"true\">(</mo><mrow><msup><mi mathvariant=\"bold-italic\">w</mi><mi>T</mi></msup><msub><mi mathvariant=\"bold-italic\">x</mi><mi mathvariant=\"bold-italic\">i</mi></msub><mo>+</mo><mi>b</mi></mrow><mo stretchy=\"true\">)</mo></mstyle><mo mathvariant=\"italic\">&#x2265;</mo><mn mathvariant=\"italic\">1</mn><mo mathvariant=\"italic\">-</mo><msub><mi>&#x3BE;</mi><mi mathvariant=\"italic\">i</mi></msub><mo mathvariant=\"italic\">&#xA0;</mo><mo mathvariant=\"italic\">&#xA0;</mo><mo mathvariant=\"italic\">&#xA0;</mo><mo mathvariant=\"italic\">&#xA0;</mo><mi>i</mi><mo mathvariant=\"italic\">=</mo><mn mathvariant=\"italic\">1</mn><mo mathvariant=\"italic\">,</mo><mo mathvariant=\"italic\">.</mo><mo mathvariant=\"italic\">.</mo><mo mathvariant=\"italic\">.</mo><mo mathvariant=\"italic\">,</mo><mi>N</mi></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (2) |

where the positive slack variables  are introduced and penalised in the objective function to allow for a subset of points to be misclassified. The value of the hyperparameter *C* is important for a good generalisation capability. The featuresmay also be mapped onto a higher-dimensional space via a feature map {"mathml":"<math style=\"font-family:stix;font-size:14px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"14px\"><mi mathvariant=\"bold-italic\">&#x3C6;</mi></mstyle></math>","origin":"MathType for Microsoft Add-in"} which can be a linear kernel for linear classification problems, a radial basis function kernel (RBF) or a polynomial kernel for nonlinear classification problems:

|  |  |
| --- | --- |
| {"mathml":"<math style=\"font-family:stix;font-size:14px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"14px\"><msub><mi>K</mi><mi>rbf</mi></msub><mfenced><mrow><msub><mi mathvariant=\"bold-italic\">x</mi><mi>i</mi></msub><mo>,</mo><msub><mi mathvariant=\"bold-italic\">x</mi><mi>j</mi></msub></mrow></mfenced><mo>=</mo><mi mathvariant=\"bold-italic\">&#x3C6;</mi><mrow><mo mathvariant=\"bold\" stretchy=\"true\">(</mo><msub><mi mathvariant=\"bold-italic\">x</mi><mi>i</mi></msub><mo mathvariant=\"bold\" stretchy=\"true\">)</mo></mrow><mi mathvariant=\"bold-italic\">&#x3C6;</mi><mrow><mo mathvariant=\"bold\" stretchy=\"true\">(</mo><msub><mi mathvariant=\"bold-italic\">x</mi><mi>j</mi></msub><mo mathvariant=\"bold\" stretchy=\"true\">)</mo></mrow><mo>=</mo><mfenced open=\"&lt;\" close=\"&gt;\"><mrow><msub><mi mathvariant=\"bold-italic\">x</mi><mi>i</mi></msub><mo>,</mo><msub><mi mathvariant=\"bold-italic\">x</mi><mi>j</mi></msub></mrow></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (3) |
| {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msub><mi>K</mi><mi>rbf</mi></msub><mfenced><mrow><msub><mi mathvariant=\"bold-italic\">x</mi><mi>i</mi></msub><mo>,</mo><msub><mi mathvariant=\"bold-italic\">x</mi><mi>j</mi></msub></mrow></mfenced><mo>=</mo><mi mathvariant=\"bold-italic\">&#x3C6;</mi><mrow><mo mathvariant=\"bold\" stretchy=\"true\">(</mo><msub><mi mathvariant=\"bold-italic\">x</mi><mi>i</mi></msub><mo mathvariant=\"bold\" stretchy=\"true\">)</mo></mrow><mi mathvariant=\"bold-italic\">&#x3C6;</mi><mrow><mo mathvariant=\"bold\" stretchy=\"true\">(</mo><msub><mi mathvariant=\"bold-italic\">x</mi><mi>j</mi></msub><mo mathvariant=\"bold\" stretchy=\"true\">)</mo></mrow><mo>=</mo><mi>exp</mi><mfenced><mrow><mo>-</mo><mi mathvariant=\"normal\">&#x3B3;</mi><msup><mfenced open=\"|\" close=\"|\"><mrow><msub><mi mathvariant=\"bold-italic\">x</mi><mi>i</mi></msub><mo>-</mo><msub><mi mathvariant=\"bold-italic\">x</mi><mi>j</mi></msub></mrow></mfenced><mn>2</mn></msup></mrow></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (4) |
| {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msub><mi>K</mi><mi>poly</mi></msub><mfenced><mrow><msub><mi mathvariant=\"bold-italic\">x</mi><mi>i</mi></msub><mo>,</mo><msub><mi mathvariant=\"bold-italic\">x</mi><mi>j</mi></msub></mrow></mfenced><mo>=</mo><mi mathvariant=\"bold-italic\">&#x3C6;</mi><mrow><mo mathvariant=\"bold\" stretchy=\"true\">(</mo><msub><mi mathvariant=\"bold-italic\">x</mi><mi>i</mi></msub><mo mathvariant=\"bold\" stretchy=\"true\">)</mo></mrow><mi mathvariant=\"bold-italic\">&#x3C6;</mi><mrow><mo mathvariant=\"bold\" stretchy=\"true\">(</mo><msub><mi mathvariant=\"bold-italic\">x</mi><mi>j</mi></msub><mo mathvariant=\"bold\" stretchy=\"true\">)</mo></mrow><mo>=</mo><msup><mfenced><mrow><mi mathvariant=\"normal\">&#x3B3;</mi><msub><mi mathvariant=\"bold-italic\">x</mi><mi>i</mi></msub><mo mathvariant=\"bold-italic\">&#xB7;</mo><msub><mi mathvariant=\"bold-italic\">x</mi><mi>j</mi></msub><mo mathvariant=\"italic\">+</mo><mi>r</mi></mrow></mfenced><mi>d</mi></msup></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (5) |

where , *r* and *d* are additional hyperparameters. The solution of the convex optimization problem (2) provides optimal values ***w\**** and *b\** as:

|  |  |
| --- | --- |
| {"mathml":"<math style=\"font-family:stix;font-size:14px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"14px\"><msup><mi mathvariant=\"bold-italic\">w</mi><mo>*</mo></msup><mo>=</mo><mstyle displaystyle=\"false\"><munderover><mo>&#x2211;</mo><mrow><mi>i</mi><mo>=</mo><mn>1</mn></mrow><mi>N</mi></munderover></mstyle><msubsup><mi>a</mi><mi>i</mi><mo>*</mo></msubsup><msub><mi>y</mi><mi>i</mi></msub><mi>&#x3C6;</mi><mfenced><msub><mi mathvariant=\"bold-italic\">x</mi><mi>i</mi></msub></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (6) |
| {"mathml":"<math style=\"font-family:stix;font-size:14px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"14px\"><msup><mi>b</mi><mo>*</mo></msup><mo>=</mo><msub><mi>y</mi><mi>j</mi></msub><mstyle displaystyle=\"false\"><mo>-</mo><munderover><mo>&#x2211;</mo><mrow><mi>i</mi><mo>=</mo><mn>1</mn></mrow><mi>N</mi></munderover></mstyle><msubsup><mi>a</mi><mi>i</mi><mo>*</mo></msubsup><msub><mi>y</mi><mi>i</mi></msub><mi>K</mi><mfenced><mrow><msub><mi mathvariant=\"bold-italic\">x</mi><mi>i</mi></msub><mo>,</mo><msub><mi mathvariant=\"bold-italic\">x</mi><mi mathvariant=\"italic\">j</mi></msub></mrow></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (7) |

where  denotes the optimal Lagrange multiplier for the *i*th constraint and the label  is chosen such that .The features  for whic are those lying on the classification boundary and are called support vectors. Finally, the predicted class of a new sample ***x*** is given by:

|  |  |
| --- | --- |
| {"mathml":"<math style=\"font-family:stix;font-size:14px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"14px\"><mi>f</mi><mfenced><mi mathvariant=\"bold-italic\">x</mi></mfenced><mo>=</mo><mi>sgn</mi><mfenced><mover><mi>y</mi><mo>^</mo></mover></mfenced><mo>=</mo><mi>sgn</mi><mfenced><mstyle displaystyle=\"false\"><munder><mo>&#x2211;</mo><mrow><mi>i</mi><mo mathvariant=\"italic\">&#x2208;</mo><msup><mi>N</mi><mi>s</mi></msup></mrow></munder><msubsup><mi>a</mi><mi>i</mi><mo>*</mo></msubsup><msub><mi>y</mi><mi>i</mi></msub><mi>K</mi><mfenced><mrow><msub><mi mathvariant=\"bold-italic\">x</mi><mi>i</mi></msub><mo>,</mo><mi mathvariant=\"bold-italic\">x</mi></mrow></mfenced><mo>+</mo><msup><mi>b</mi><mo>*</mo></msup></mstyle></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (8) |

where the predicted class {"mathml":"<math style=\"font-family:stix;font-size:14px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"14px\"><mover><mi>y</mi><mo>^</mo></mover></mstyle></math>","origin":"MathType for Microsoft Add-in"} of a new sample ***x*** is only related to the support vectors  whose subscripts are denoted by *Ns*, along with the corresponding Lagrange multipliers *ai*\*.

* + 1. ANN formulation

ANNs are well-known for their ability to approximate complex nonlinear functions. In the context of binary classification, ANNs can provide a probability score for belonging to either of the classes using a softmax function. Given the weight matrix ***W****k*-1 and bias vector***B****k*-1, the output ***m****k* of any neurons in hidden layer *k* can be calculated as:

|  |  |
| --- | --- |
| {"mathml":"<math style=\"font-family:stix;font-size:14px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"14px\"><msub><mi mathvariant=\"bold-italic\">m</mi><mi>k</mi></msub><mo>=</mo><msub><mi>&#x3C3;</mi><mi>k</mi></msub><mfenced><msub><mi mathvariant=\"bold-italic\">l</mi><mi>k</mi></msub></mfenced><mo>=</mo><msub><mi>&#x3C3;</mi><mi>k</mi></msub><mfenced><mrow><msub><mi mathvariant=\"bold-italic\">W</mi><mrow><mi>k</mi><mo>-</mo><mn>1</mn></mrow></msub><msub><mi mathvariant=\"bold-italic\">m</mi><mrow><mi>k</mi><mo>-</mo><mn>1</mn></mrow></msub><mo>+</mo><msub><mi mathvariant=\"bold-italic\">B</mi><mrow><mi>k</mi><mo>-</mo><mn>1</mn></mrow></msub></mrow></mfenced><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#x2200;</mo><mi>k</mi></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (9) |

where {"mathml":"<math style=\"font-family:stix;font-size:14px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"14px\"><msub><mi mathvariant=\"bold-italic\">l</mi><mi>k</mi></msub><mo>=</mo><msub><mi mathvariant=\"bold-italic\">W</mi><mrow><mi>k</mi><mo>-</mo><mn>1</mn></mrow></msub><msub><mi mathvariant=\"bold-italic\">m</mi><mrow><mi>k</mi><mo>-</mo><mn>1</mn></mrow></msub><mo>+</mo><msub><mi mathvariant=\"bold-italic\">B</mi><mrow><mi>k</mi><mo>-</mo><mn>1</mn></mrow></msub></mstyle></math>","origin":"MathType for Microsoft Add-in"}. The choice of activation function {"mathml":"<math style=\"font-family:stix;font-size:14px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"14px\"><msub><mi mathvariant=\"bold-italic\">&#x3C3;</mi><mi>k</mi></msub></mstyle></math>","origin":"MathType for Microsoft Add-in"} can have a great influence on the ability to capture nonlinear behaviour. Here, the activation functions tanh and relu are investigated for the hidden layers. The sigmoid function is applied to the output layer to rescale the predictions within [0,1]. These functions are given by:

|  |  |
| --- | --- |
| {"mathml":"<math style=\"font-family:stix;font-size:14px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"14px\"><msub><mi mathvariant=\"bold-italic\">&#x3C3;</mi><mi>tanh</mi></msub><mfenced><msub><mi mathvariant=\"bold-italic\">l</mi><mi>k</mi></msub></mfenced><mo mathvariant=\"italic\">=</mo><mn>1</mn><mo mathvariant=\"italic\">-</mo><mfrac><mn>2</mn><mrow><mi>exp</mi><mrow><mo mathvariant=\"italic\" stretchy=\"true\">(</mo><mn>2</mn><msub><mi mathvariant=\"bold-italic\">l</mi><mi>k</mi></msub><mo mathvariant=\"italic\" stretchy=\"true\">)</mo></mrow><mo>+</mo><mn>1</mn></mrow></mfrac></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (10) |
| {"mathml":"<math style=\"font-family:stix;font-size:14px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"14px\"><msub><mi mathvariant=\"bold-italic\">&#x3C3;</mi><mi>relu</mi></msub><mfenced><msub><mi mathvariant=\"bold-italic\">l</mi><mi>k</mi></msub></mfenced><mo mathvariant=\"italic\">=</mo><mi>max</mi><mfenced><mrow><mn>0</mn><mo>,</mo><msub><mi mathvariant=\"bold-italic\">l</mi><mi>k</mi></msub></mrow></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (11) |
| {"mathml":"<math style=\"font-family:stix;font-size:14px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"14px\"><msub><mi mathvariant=\"bold-italic\">&#x3C3;</mi><mi>sigmoid</mi></msub><mfenced><msub><mi mathvariant=\"bold-italic\">l</mi><mi>k</mi></msub></mfenced><mo mathvariant=\"italic\">=</mo><mfrac><mn>1</mn><mrow><mn>1</mn><mo>+</mo><mi>exp</mi><mfenced><mrow><mo>-</mo><msub><mi mathvariant=\"bold-italic\">l</mi><mi>k</mi></msub></mrow></mfenced></mrow></mfrac></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (12) |

A loss function *L* is then minimised to train the ANNs to match the predicted class  of the samples {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>i</mi><mo>=</mo><mn>1</mn><mo>,</mo><mo>.</mo><mo>.</mo><mo>.</mo><mo>,</mo><mi>N</mi></mstyle></math>","origin":"MathType for Microsoft Add-in"}:

|  |  |
| --- | --- |
| {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><munder><mi>min</mi><mrow><mi mathvariant=\"bold-italic\">W</mi><mo mathvariant=\"bold-italic\">,</mo><mi mathvariant=\"bold-italic\">B</mi></mrow></munder><mo>=</mo><mo>-</mo><mfrac><mn>1</mn><mi>N</mi></mfrac><mstyle displaystyle=\"false\"><munderover><mo>&#x2211;</mo><mrow><mi>i</mi><mo>=</mo><mn>1</mn></mrow><mi>N</mi></munderover></mstyle><msub><mi>y</mi><mi>i</mi></msub><mi>ln</mi><mfenced><msub><mover><mi>y</mi><mo>^</mo></mover><mi>i</mi></msub></mfenced><mo>+</mo><mfenced><mrow><mn>1</mn><mo>-</mo><msub><mi>y</mi><mi>i</mi></msub></mrow></mfenced><mi>ln</mi><mfenced><mrow><mn>1</mn><mo>-</mo><msub><mover><mi>y</mi><mo>^</mo></mover><mi>i</mi></msub></mrow></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (13) |

* 1. Methodology
     1. Data generation

Classifier surrogates are constructed to predict the phase stability of a liquid phase consisting of a solute (ibuprofen) and a solvent pair, where and are chosen in a set *S*. Two cases are investigated: In Case 1, the models are trained and used at a fixed temperature of 300 K (Jonuzaj et al., 2016). In Case 2, phase stability is considered over the temperature range 293.15-318.15 K, in which the mixtures exhibit solid-liquid-(liquid) equilibria (Watson et al., 2021). Data are generated from phase equilibria calculations in phasepy (Chaparro and Mejía, 2020) using UNIFAC (Fredenslund et al., 1975). The sizes of the datasets for both Cases are described in Table 1. The solvents considered in Case 1 are methanol, ethanol, 2-propanol, acetone, MIBK, ethylacetate, chloroform, toluene, and water; and in Case 2, water, n-pentane, n-heptane, ethanol, 1-propanol, 1-butanol, 1-pentanol, and acetone. The van der Waals surface area and volume parameters, *qi* and *ri,* are used to characterize each solvent. The total mole fractions *zi* are furthermore included to represent the mixture. Thus (*qa,qb,ra,rb,za,zb*) and (*qa,qb,ra,rb,za,zb,T*) are used as inputs in Cases 1 and 2, respectively.

Table.1. Description of generated datasets for both case studies

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | No. solvents | No. solvent pairs | No. samples | Stable: Non-stable | Generation time /s |
| 1 | 9 | 36 | 18,427 | 86.62%:13.38% | 659.98 |
| 2 | 8 | 56 | 228,976 | 82.96%:17.04% | 7018.74 |

* + 1. Evaluation of the classifier models

Since the dataset is quite imbalanced, the Matthew’s Correlation Coefficient (MCC) (Chicco and Jurman, 2020) is used as the metric for all the models:

|  |  |
| --- | --- |
| {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>MCC</mi><mo>=</mo><mfrac><mrow><msub><mi mathvariant=\"italic\">P</mi><mrow><mi mathvariant=\"italic\">T</mi><mi mathvariant=\"italic\">P</mi></mrow></msub><msub><mi mathvariant=\"italic\">P</mi><mrow><mi mathvariant=\"italic\">T</mi><mi mathvariant=\"italic\">N</mi></mrow></msub><mo mathvariant=\"italic\">-</mo><msub><mi mathvariant=\"italic\">P</mi><mrow><mi mathvariant=\"italic\">F</mi><mi mathvariant=\"italic\">P</mi></mrow></msub><msub><mi mathvariant=\"italic\">P</mi><mrow><mi mathvariant=\"italic\">F</mi><mi mathvariant=\"italic\">N</mi></mrow></msub></mrow><msqrt><mstyle mathvariant=\"italic\"><mo stretchy=\"true\">(</mo><mrow><msub><mi>P</mi><mrow><mi>T</mi><mi>P</mi></mrow></msub><mo>+</mo><msub><mi>P</mi><mrow><mi>F</mi><mi>P</mi></mrow></msub></mrow><mo stretchy=\"true\">)</mo></mstyle><mstyle mathvariant=\"italic\"><mo stretchy=\"true\">(</mo><mrow><msub><mi>P</mi><mrow><mi>T</mi><mi>P</mi></mrow></msub><mo>+</mo><msub><mi>P</mi><mrow><mi>F</mi><mi>N</mi></mrow></msub></mrow><mo stretchy=\"true\">)</mo></mstyle><mstyle mathvariant=\"italic\"><mo stretchy=\"true\">(</mo><mrow><msub><mi>P</mi><mrow><mi>T</mi><mi>N</mi></mrow></msub><mo>+</mo><msub><mi>P</mi><mrow><mi>F</mi><mi>P</mi></mrow></msub></mrow><mo stretchy=\"true\">)</mo></mstyle><mstyle mathvariant=\"italic\"><mo stretchy=\"true\">(</mo><mrow><msub><mi>P</mi><mrow><mi>T</mi><mi>N</mi></mrow></msub><mo>+</mo><msub><mi>P</mi><mrow><mi>F</mi><mi>N</mi></mrow></msub></mrow><mo stretchy=\"true\">)</mo></mstyle></msqrt></mfrac></mstyle></math>","origin":"MathType for Microsoft Add-in"} | (14) |

where *PTP* (true positive) and *PTN* (true negative) denote the number of samples correctly classified as positive (stable) and negative (non-stable), respectively; and *PFP* (false positive) and *PFN* (false negative) denote the number of samples incorrectly classified as positive and negative, respectively. The MCC score takes values from -1 to +1, where -1 indicates only false predictions, 0 random predictions, and +1perfect predictions.

* + 1. Hyperparameter tuning and model training

Hyperparameter tuning is conducted for each model via Optuna(Akiba et al., 2019). For ANNs, a single hidden layer is used in Case 1, and 3 hidden layers in Case 2. The numbers of neurons are set by Optuna. The dataset is split 80:20 into a training dataset and a test dataset and five-fold cross-validation is conducted on the training dataset, using stratified sampling to ensure consistent distributions of stable and non-stable mixtures in each fold. The selection of hyperparameters is based on achieving the highest average MCC score on the validation datasets. Subsequently, the classifier models are retrained using these optimized hyperparameters. The SVMs are built on scikit-learn v1.3.2 (Pedregosa et al., 2011) and ANNs are built on PyTorch v2.1.0 (Paszke et al., 2019)*.* All the models are trained on Windows11 using Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz with 64GB of RAM and NVIDIA RTX A4000 GPU with 16GB of RAM. The GPU is only used in training ANNs.

Table.2. Training and testing results for SVMs

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | Kernel | MCCtraining | MCCtest | Training time /s | No. support vectors |
| 1 | Linear | 0.8990 | 0.8914 | 2.12 | 968 |
| RBF | 0.9947 | ***0.9894*** | ***0.20*** | 183 |
| Poly-2 | 0.9708 | 0.9720 | 0.84 | 354 |
| Poly-3 | 0.9933 | 0.9848 | 1.38 | 98 |
| Poly-4 | ***0.9994*** | 0.9871 | 3.47 | ***54*** |
| 2 | Linear | 0.3798 | 0.3776 | 774.49 | 64,297 |
| RBF | ***0.9912*** | ***0.9864*** | ***61.19*** | 4,247 |
| Poly-2 | 0.9517 | 0.9504 | 468.64 | 8,195 |
| Poly-3 | 0.9735 | 0.9705 | 655.62 | 4,275 |
| Poly-4 | 0.9840 | 0.9813 | 746.42 | ***2,839*** |

* 1. Results and discussion

The performance of all models is presented in Tables 2 (SVMs) and 3 (ANNs). Polynomial SVMs are denoted by Poly-*d* where *d* denotes the degree. Amongst SVMs, the linear SVM performs worst, with an MCC score of 0.38 in Case 2. The linear SVMs require the largest number of support vectors but this has no impact on model complexity of the decision function. However, this property does not hold for other nonlinear kernels. The RBF-SVM performs quite well in both cases, achieving an MCC score of 0.99 with the shortest training time. It also requires fewer support vectors than Poly-2 in both cases. The performance of polynomial SVMs improves as the degree increases. In Case 1, Poly-4 exhibits a performance comparable to that of RBF, while in Case 2 it requires a larger training time.

With ANNs, the tanh activation function generates models with a higher MCC score than relu. In Case 1, the tanh-ANN has a classification performance comparable to that of the Poly-4 SVM, while the relu-ANN performs similarly to the RBF-SVM. In Case 2, the tanh-ANN is slightly worse than the RBF-SVM, but better than other SVMs. The performance of the relu-ANN is close to that of the Poly-4 SVM. Adding more layers to the ANNs can improve the MCC score, but it does also increase model complexity, especially considering that the training time is already much longer than for SVMs in both cases. One advantage of ANNs over SVNs is that they return a probability of the phase being stable, which is more intuitive than the distance between the input vectors and the classification boundary.

Table.3. Training results for ANNs

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | Activation function | MCCtraining | MCCtest | Training time /s | ANN structure  (Nodes per layer) |
| 1 | tanh | **0.9977** | **0.9906** | **84.19** | {6,15,1} |
| relu | 0.9947 | 0.9871 | 165.55 | {6,17,1} |
| 2 | tanh | **0.9883** | **0.9848** | 1,394.84 | {7,19,17,1,1} |
| relu | 0.9814 | 0.9784 | **1,096.46** | {7,14,15,8,1} |

Fig. 1 illustrates the performance of RBF-SVM with two phase diagrams for the ibuprofen-ethanol-water mixture from Case 2. Here, the larger the value {"mathml":"<math style=\"font-family:stix;font-size:14px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"14px\"><mover><mi>y</mi><mo>^</mo></mover></mstyle></math>","origin":"MathType for Microsoft Add-in"} returned by the SVM at a given composition and temperature, the more likely the liquid phase to be miscible, and vice versa. As the temperature increases, the regions of immiscibility (liquid-liquid and solid-liquid-liquid). The RBF-SVM captures this phenomenon with some uncertainty around the exact position of the phase boundary, as shown in dark blue. Similar performance of this classifier is observed for all mixtures, including some mixtures which are always miscible. The typical CPU time to evaluate the miscibility of one composition is  seconds and is much faster  seconds).

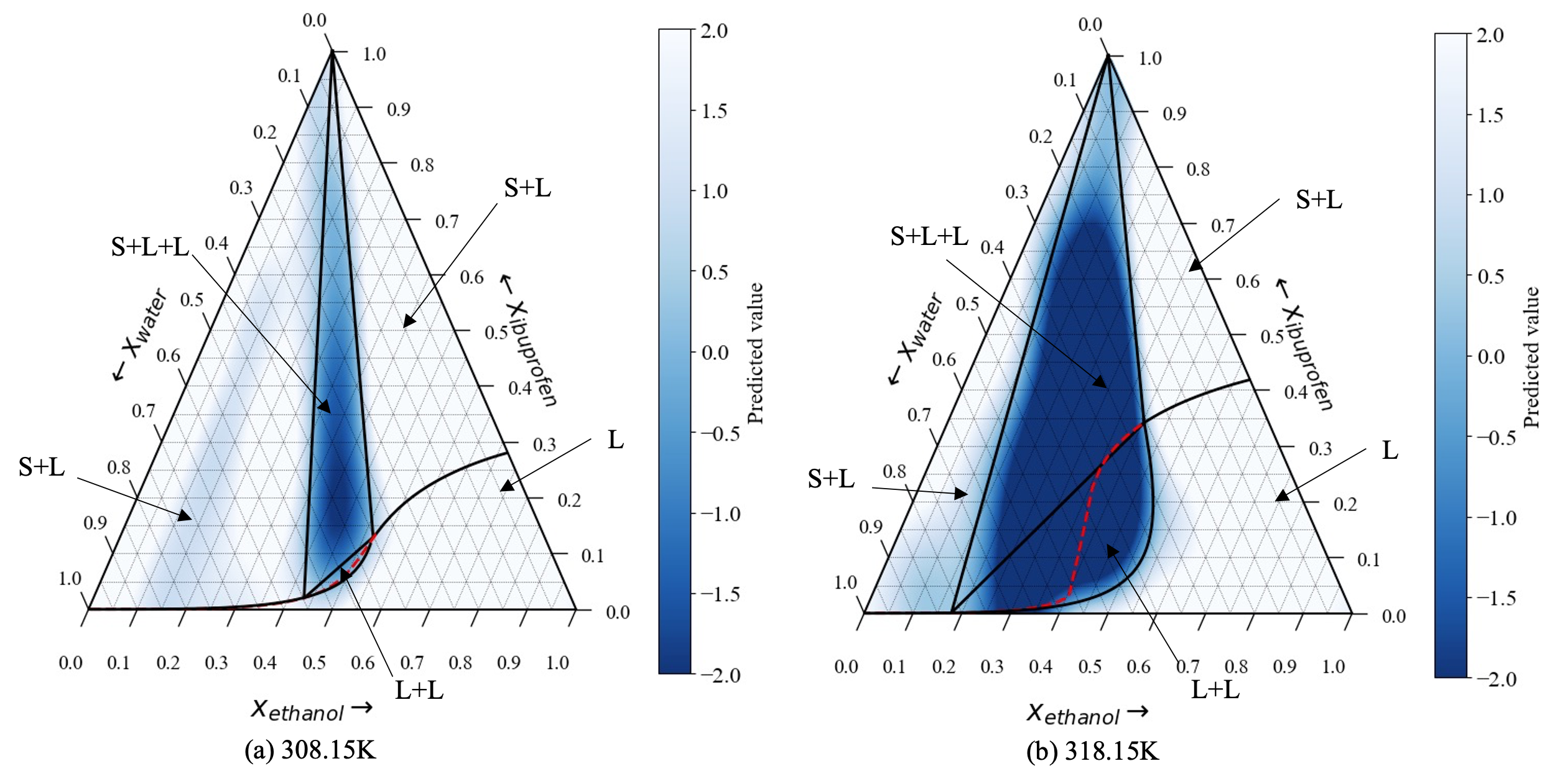


Figure 1. Phase diagram for Ibuprofen-Ethanol-Water mixture at (a) 308.15K and (b) 318.15K. The solid curves represent the phase boundaries, while the dashed curve represents a metastable solid-liquid boundary. The color scales represent the magnitude of the RBF-SVM predictions {"mathml":"<math style=\"font-family:stix;font-size:10px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"10px\"><mover><mi>y</mi><mo>^</mo></mover></mstyle></math>","origin":"MathType for Microsoft Add-in"} in Eq (8).

* 1. Conclusion

This paper investigated the potential application of both surrogate SVMs and ANNs to predict the phase stability of multicomponent mixtures. Based on two case studies, several such surrogates achieved good performance with RBF-SVM performing best overall in both MCC score and training time. In addition, considering the number of support vectors, RBF-SVM was found to offer good potential to be embedded into an optimisation problem without increasing the problem scale significantly. In future work, these models will be used to replace the phase stability constraints in optimisation problems such as solvent design and to investigate the impact on optimization efficiency and solution quality.

Acknowledgement

We gratefully acknowledge funding from the UK Engineering and Physical Sciences Research Council (EPSRC) under grant EP/W003317/1.

Data Statement

Data supporting this article is available by writing to the corresponding author.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M., 2019. Optuna: A Next-generation Hyperparameter Optimization Framework, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD ’19. Association for Computing Machinery, New York, NY, USA, pp. 2623–2631.

Baker, L.E., Pierce, A.C., Luks, K.D., 1982. Gibbs Energy Analysis of Phase Equilibria. Society of Petroleum Engineers Journal 22, 731–742. https://doi.org/10.2118/9806-PA

Chaparro, G., Mejía, A., 2020. Phasepy: A Python based framework for fluid phase equilibria and interfacial properties computation. Journal of Computational Chemistry 41, 2504–2526.

Chicco, D., Jurman, G., 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics 21, 6.

Cortes, C., Vapnik, V., 1995. Support-vector networks. Mach Learn 20, 273–297.

Fredenslund, A., Jones, R.L., Prausnitz, J.M., 1975. Group-contribution estimation of activity coefficients in nonideal liquid mixtures. AIChE Journal 21, 1086–1099.

Gaganis, V., Varotsis, N., 2012. Non-iterative phase stability calculations for process simulation using discriminating functions. Fluid Phase Equilibria 314, 69–77.

Jonuzaj, S., Akula, P.T., Kleniati, P.-M., Adjiman, C.S., 2016. The formulation of optimal mixtures with generalized disjunctive programming: A solvent design case study. AIChE Journal 62, 1616–1633. https://doi.org/10.1002/aic.15122

Lopez-Ramirez, E., Lopez-Zamora, S., Escobedo, S., de Lasa, H., 2023. Artificial Neural Networks (ANNs) for Vapour-Liquid-Liquid Equilibrium (VLLE) Predictions in N-Octane/Water Blends. Processes 11, 2026. https://doi.org/10.3390/pr11072026

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. https://doi.org/10.48550/arXiv.1912.01703

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., 2011. Scikit-learn: Machine learning in Python. the Journal of machine Learning research 12, 2825–2830.

Schmitz, J.E., Zemp, R.J., Mendes, M.J., 2006. Artificial neural networks for the solution of the phase stability problem. Fluid Phase Equilibria, Proceedings of the Seventeenth European Conference on Thermophysical Properties 245, 83–87.

Watson, O.L., Jonuzaj, S., McGinty, J., Sefcik, J., Galindo, A., Jackson, G., Adjiman, C.S., 2021. Computer Aided Design of Solvent Blends for Hybrid Cooling and Antisolvent Crystallization of Active Pharmaceutical Ingredients. Org. Process Res. Dev. 25, 1123–1142.