Applied Physics B (2021) 127:34
https://doi.org/10.1007/s00340-021-07582-0

Applied Physics B
Lasers and Optics

Evaluation of machine learning methods for classification of rotational absorption spectra for gases in the 220–330 GHz range

M. Arshad Zahangir Chowdhury[1] · Timothy E. Rice[1] · Matthew A. Oehlschlaeger[1]

Received: 20 October 2020 / Accepted: 30 January 2021 / Published online: 16 February 2021

Abstract

Machine learning (ML) methods are implemented to classify rotational absorption spectra for gas-phase compounds in the THz region, specifically 220–330 GHz where experimental data is available. Eight ML methods were trained in both standard and one-versus-rest (OVR) implementations using simulated absorption spectra for 12 volatile organic compounds and halogenated hydrocarbons of interest in industrial and environmental gas sensing applications. The performance of the resulting ML classifiers was compared against simulated training spectra in both a 70–30 training–testing split and in tenfold cross-validation studies, with the classifiers exhibiting accuracies in the range of 88–99% for simulated spectra. The classifiers were then tested for their ability to classify noisy experimental rotational spectra for methanol, ethanol, formic acid, acetaldehyde, acetonitrile, and chloromethane. The OVR implementations of the support vector machine (SVM) classifier with both linear and radial basis function kernels and the multi-layer perceptron (MLP) classifier achieved average classification accuracies of 87–94% for the experimental dataset. The study shows that THz spectra in the present frequency region provide a sufficient spectral fingerprint for ML classifiers to learn and predict speciation, allowing automated gas sensing. The present methods can be extrapolated to different frequency ranges and compounds and conditions.

1 Introduction

Absorption spectroscopy is commonly applied for the non-intrusive identification of gas-phase species and quantitative determination of their concentrations in industrial, environmental, and research settings [1]. Absorption measurements in unknown single- or multi-component gases are often complicated by the vast number of spectral features (transitions) present within an experimental frequency range, depending on the complexity of the molecules electronic (ultraviolet to visible), vibrational (infrared), or rotational (terahertz to microwave) energy level structure. In fact, the complexity of molecular vibrational and rotational spectroscopy is so great, that measurement of these features often provides a distinct "fingerprint" for a probed species. However, like human fingerprints, identification of the species responsible for a measured spectrum is usually not trivial. Significant effort can be required to assign experimentally observed features, using databases of spectroscopic transitions based on theory and experiments, to determine speciation. Further, spectral fitting or modeling is usually required to make quantitative measurements of gas-phase concentrations [2].

Absorption spectroscopy is well established in the literature [2–6]. In the terahertz (THz) wave region (0.1–10 THz), where our group has been recently developing gas-phase absorption sensors [7], molecules absorb radiation due to changes in their quantized rotational energy levels. The THz wave region can have several advantages over the commonly employed infrared region for absorption spectroscopy gas sensing. Polar gas molecules offer strong and distinct spectra in the THz wave region, often allowing greater sensitivity and selectivity than other frequency bands. THz waves are also not as susceptible to scattering or extinction from particles, as is infrared radiation, and large regions of the THz exist with no interfering absorption from water vapor. Additionally, THz waves can be generated using microelectronic sources [8, 9] and, hence, robust, miniature, and inexpensive gas sensors can be developed in this frequency range [10].

The determination of speciation from a measured absorption spectrum is a classification problem, suitable for supervised machine learning (ML) approaches. Supervised ML methods learn rules or functions from training observations

✉ M. Arshad Zahangir Chowdhury
chowdm@rpi.edu

[1] Department of Mechanical, Aerospace, and Nuclear Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA

(e.g., spectra) and, when they encounter a new observation, assign the new observation to a particular category or class based on the learned rules or functions. Furthermore, ML algorithms are capable of recognizing complex patterns in high-dimensional data, ideal for recognizing features in complex rotational or vibrational fingerprint spectra.

ML methods have been widely used for various problems involving materials classification from experimental characterization data, including, the classification of solid samples such as coal from proximate analysis data [11], wood from laser-induced breakdown spectra [12], and heavy minerals from scanning electron microscopy images [13]. Laser-induced breakdown spectroscopy measurements have been used to train ML models to classify olive oils [14, 15] and Biodiesels [16] have also been classified using measured near-infrared spectra with ML methods. However, studies related to the identification of gas-phase species from non-intrusive absorption spectroscopy using ML classifiers are few in the literature, although ML has been used in addressing a variety of problems relevant to spectroscopy and environmental monitoring [17–24].

For gas classification, in recent years, deep neural networks (DNN) have been leveraged [25]. DNNs offer potentially high classification accuracy via in-built feature learning, at the cost of optimizing millions of parameters which requires a large amount of training data. Furthermore, when compared to conventional ML algorithms, DNNs require long training times [25]. While training a DNN algorithm can be accomplished offline, the cost of retraining, to update an existing gas sensing model, and the lack of interpretability of DNNs are two of the method's biggest drawbacks. Small and fully connected neural networks, known as multi-layer perceptrons (MLPs), require fewer parameters and less training time than DNNs and offer reduced model complexity while handling nonlinearity and high-dimensionality in training data, and acceptable classification accuracy [26]. MLPs have been integrated with gas sensor array outputs to recognize a variety of simple gases species [25, 26].

Support vector machines (SVM) are popular ML classifiers due to their optimal decision boundary identification capabilities [27, 28] and have been used for gas classification [29, 30] and determination of gas concentrations for mixture components [31]. SVMs are developed using statistical learning theory and use training samples closest to the boundary, known as support vectors, to find optimum hyperplanes for the construction of decision boundaries. SVM performance can be improved using kernel functions [23] and using a penalizing hyperparameter to solve soft-margin classification problems, where classes are inherently not separable, with reduced error [28]. Other mature algorithms such as k-nearest neighbors (k-NN) [32], decision trees (DT) [33], random forest (RF) [34], and boosting methods [35, 36] have also been employed in gas classification problems involving gas sensor or electronic nose data. However, tree-based methods often suffer from "overfitting" which can drastically reduce their performance [37].

The shape of an absorption spectra for a given chemical species originating within a frequency band is mostly unique [37, 38]. The unique spectral shape for many polar molecules in the THz region [39] can form the basis of learning for ML classifiers. While the spectral fingerprint in any frequency domain will vary depending on the thermodynamic conditions, absorption path length, the concentration of the absorbing molecule, and composition of the bath gas, the frequencies of transitions and overall shape of the spectral fingerprint for a particular compound will remain generally intact, or self-similar, and ML methods can be trained to identify spectral fingerprints, and provide an automated identification of species. Motivated by the pattern recognition and high-dimensional data capabilities of ML classifiers, combined with superior selectivity [7, 40, 41] due to unique fingerprint available in the 220–330 GHz frequency range (7.33–11 cm^{-1}) which greatly reduces human efforts in identifying features and matching patterns in spectral data for speciation, we demonstrate here that ML classifiers can be used to develop a fast spectra recognition tool to complement available spectroscopic tools.

In the present work, we investigate the potential for eight different supervised ML classification algorithms for the identification of gas-phase species based on absorption spectra in the 220–330 GHz region, where prior experimental studies have been carried out. A number of supervised ML classifiers were trained to identify spectra, namely, *k*-nearest neighbors, decision trees, random forest, support vector machine (with linear and radial basis function kernels), multi-layer perceptron, and decision trees and random forest with adaptive boosting. These classifiers were trained using two different strategies. First, in the regular implementation strategy, the classifiers were fit across all the classes/compounds. Second, the One-Vs-Rest (OVR) strategy was implemented, where a single classifier is fitted per class/label/compound.

In total, 16 different ML classifiers were trained on absorbance spectra of 12 compounds in the 220–330 GHz range. Absorption spectra were simulated for the 12 compounds using fundamental spectroscopic parameters. ML classifiers were trained and optimized using 70% of 1968 simulated spectra and then tested using the remaining 30% of simulated spectra. Tenfold stratified cross-validation on the training dataset was performed to interrogate the performance baseline and any potential weaknesses of classifiers. Laboratory measurements of absorption within the 220–330 GHz frequency range were used to further test, validate, and determine the most suitable ML classifiers for the accurate and automated identification of species.

2 Methodology

2.1 Experimental data for testing and validation

Spectral absorption measurements from prior studies carried out at Rensselaer are used to test and validate ML classifier performance. See Rice et al. [7] for details of the experimental methods. Absorption spectra in these studies were characterized for 220–330 GHz for pure volatile organic compounds (VOCs) and simple pure halogenated hydrocarbons at pressures of 0.5–16 Torr and at room temperature (297 K). Here, measurements for methanol, ethanol, formic acid, acetaldehyde, acetonitrile, and chloromethane are used for testing and validation.

In the prior work of Rice et al. [7], THz wave radiation was generated via a microelectronic-based system, which multiplies the output of a radio frequency (RF) synthesizer. The THz wave radiation was passed through a gas cell, containing the chemical species of interest, and focused onto a Schottky diode detector. See Fig. 1 for a schematic of the experimental setup and Fig. 2 for example measured spectra for three different compounds. To increase the signal-to-noise of the measurements, the RF source was amplitude modulated at high frequency and the detector signal was demodulated in a lock-in amplifier to extract the transmitted signal. Simultaneously, the RF signal was slowly swept in frequency space to measure absorption spectra over a range of 220–330 GHz. The Beer-Lambert law was used to determine the absorbance, based on the transmitted THz signal (I) and the reference signal (I_0):

$$-\ln\left(\frac{I}{I_0}\right) = \epsilon c L = A,$$

where A is the absorbance of the gas sample, ϵ is the absorption coefficient, c is the molar concentration of the gas sample, and L is the optical path length. The absorption coefficient depends on frequency, thermodynamic conditions (pressure and temperature), and gas sample concentrations (through collisional line broadening contributions).

2.2 Simulated training spectra

Supervised ML classifiers were exclusively trained using simulated absorption spectra for twelve compounds in the 220–330 GHz frequency range calculated based on fundamental spectroscopic parameters (line center frequencies, line intensities, lower state energies, broadening parameters) taken from either the HITRAN and JPL molecular spectroscopy databases as listed in Table 1. The compounds were chosen based on the availability of spectroscopic parameters, availability of absorption spectra measured in our laboratory in the frequency range of interest (220–330 GHz), and to include situations offering the potential for false-positive identification by ML classifiers, to test their relative performance. No experimental data was used in the training dataset.

Absorption spectra simulations were carried out for pure components over a frequency range of 220–330 GHz (7.33–11 cm^{-1}) at room temperature (297 K) and for a range of pressures from 0.3 to 16.5 Torr. The HITRAN Application Programming Interface (HAPI) [44] was used to carry out the spectral simulations within a Python code [45]. The HAPI code simulates absorption spectra based upon an input database of line positions, line strengths, lower state energies, line broadening parameters, and degeneracies using a Voigt profile for line shapes. The HITRAN database served as the primary source for the fundamental spectroscopic inputs for the generation of the simulated training data, as summarized in Table 1. However, in the case of formic acid, acetonitrile, ethanol, and acetaldehyde, compounds not cataloged in HITRAN within the present frequency space, spectroscopic parameters from the JPL molecular spectroscopy database [43] were used.

Simulated absorption spectra for the twelve compounds considered in the present study are shown in Fig. 3 at

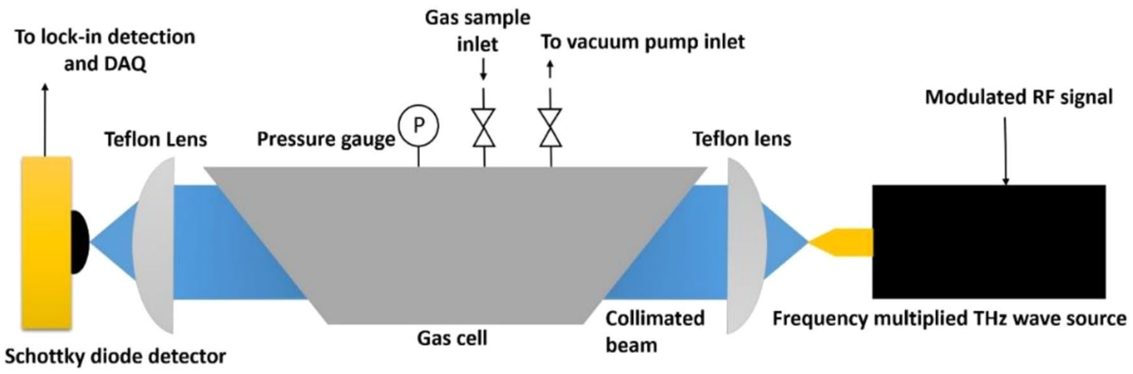


Fig. 1 Schematic of the THz absorbance spectrometer

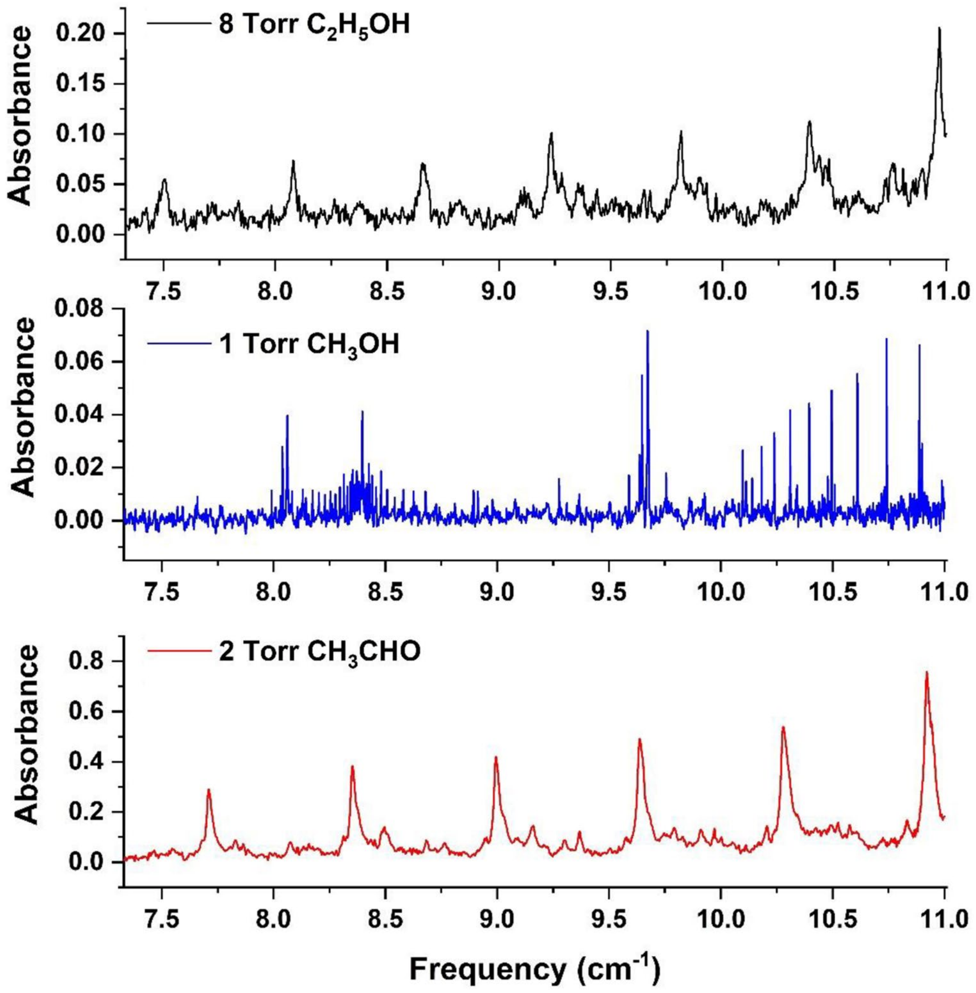


Fig. 2 Measured spectra for ethanol (upper panel), methanol (middle panel), and acetaldehyde (lower panel) at 297 K and at different pressures

Table 1 Summary of simulated training data and the source of the spectroscopic parameters used to generate the simulations

Label	Compound	Formula	Source of spectroscopic data	Experiment
0	Chloromethane	CH_3Cl	HITRAN [42]	[7]
1	Methanol	CH_3OH	HITRAN	[7]
2	Formic acid	HCOOH	JPL [43]	[7]
3	Formaldehyde	H_2CO	HITRAN	
4	Hydrogen sulfide	H_2S	HITRAN	
5	Sulfur dioxide	SO_2	HITRAN	
6	Carbonyl sulfide	OCS	HITRAN	
7	Hydrogen cyanide	HCN	HITRAN	
8	Acetonitrile	CH_3CN	JPL	[7]
9	Nitric acid	HNO_3	HITRAN	
10	Ethanol	C_2H_5OH	JPL	[7]
11	Acetaldehyde	CH_3CHO	JPL	[7]

Label indicates an integer value used as an identification index within the Python code implementation. For further information on spectroscopic parameters, see primary sources reported in [42, 43] for each compound

16.5 Torr and 297 K. In the frequency range of interest (220–330 GHz, 7.33–11 cm^{-1}), the twelve spectra are distinguished by the location and number of spectral features (lines or blended combinations of lines) and the relative absorbance for spectral features, characteristics which a ML classifier can learn to identify. While locations of spectral features for two or more compounds sometimes overlap, provided sufficient frequency range of data, there are many fingerprints for each molecule within the spectra, such that ML classifiers can learn both differences and similarities in the spectral fingerprints and their variation with pressure.

The resolution of the absorbance spectra used to train ML classifiers is important, as it provides an upper limit to the number of features available to a classifier. In building a ML classifier, all available features are never chosen to avoid overfitting. If a classifier is trained with all available simulated spectral features, the classifier will, of course, perform extremely well on the testing data. However, when presented with experimental data containing noise, the classifier may attempt to fit the noise (overfitting), leading to misclassification, or be unable to recognize

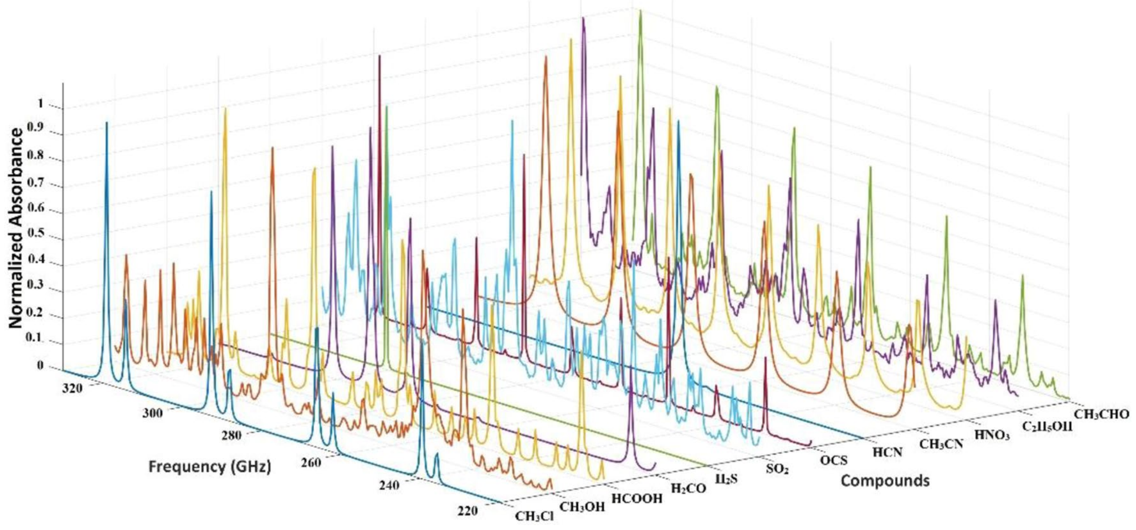


Fig. 3 Computed spectra for the 12 pure compounds that comprise the training set. Conditions: 16.5 Torr and 297 K. Absorbance value is normalized by maximum absorbance so that spectra can be visually compared

an experimental spectrum where some weak features are overcome by noise. To reduce the likelihood of overfitting, it is necessary to reduce the amount of spectral information used to train the ML classifiers, seeking to include only the data/information that provides the distinct fingerprint for each compound. In the present study, the training spectra were simulated at a spectral resolution of 0.016 cm^{-1} and more highly resolved training spectra were found to generally produce the worst preforming ML classifiers. Training spectra were simulated at 297 K for pressures from 0.3 to 16.5 Torr (0.000396–0.021749 atm) at a constant increment of 0.1 Torr (0.000131 atm), resulting 164 spectra at different pressures for each compound. Temperature was not varied in the present study, only pure single-component spectral simulations were considered, and the optical path length of 21.59 cm was used throughout the study, the experimental path length for the experimental testing data [7]; however, the path length is of no consequence to the ML classification problem, as it is a scaler multiplier within the absorbance. The entire set of training data is comprised of 1968 training spectra (twelve compounds at 164 spectra/compound) containing 229 data points/spectra (absorbance at 229 frequencies). Simulated absorption spectra were compared with experiments, where available in the literature, to verify and validate the current HAPI-based calculations, with comparisons showing good agreement. Those comparisons can be found in the appended supplementary material.

ML classifiers benefit from training data containing unique and separable features. From Fig. 3, we observe that the twelve target compounds have unique, although complicated, fingerprints; however, two compounds at the same or different conditions may have some overlapping spectral features. Hence, the uniqueness of the training data for all twelve compounds (1968 spectra) needs to be taken into account. One way to parameterize the uniqueness of the training data would be to plot pairs of features against each other, where each feature is the value of absorbance at a particular frequency. However, since the spectra contain 229 data points, it is not easy to visualize these comparisons. Hence, *t*-distributed stochastic neighbor embedding (*t*-SNE) has been used to reduce the dimensionality for visualization [46, 47]. In *t*-SNE, data points belonging to a class are assigned a location in a lower-dimensional space, such as a two- or three-dimensional space, while conserving their local structure and original clustering. A Gaussian probability distribution is constructed over pairs of spectra by the *t*-SNE algorithm such that similar high-dimensional spectra receive a high-probability value and dissimilar spectra receive a low-probability value. Afterwards, the *t*-SNE algorithm minimizes a cost function, here Kullback–Leibler divergence, using a similarity metric and by constructing a *t*-distribution in a lower-dimensional space for each mapped spectrum. The ingenuity of the stochastic neighbor embedding method, originally conceived by Hinton and Roweis [46], is that a high-dimensional object, in our case a spectrum, is represented in the lower-dimensional space by a point and the magnitude of the spacing between two points represents the relative uniqueness of those two spectra. For example, spectra of a single compound at two slightly different pressures will be very similar and will result in points on the *t*-SNE graph that are close to one another. Conversely, points for two vastly different spectra will be widely spaced. The *t*-distribution helps spread the mapped points in the lower-dimensional space. The *t*-SNE visualization reveals a complex and overlapping structures in data while preserving the original clustering.

t-SNE has been implemented on the present training dataset with the results shown in Fig. 4. The implementation details of *t*-SNE can be found in the work of Maaten and Hinton [47]. The separation of points in Fig. 4 shows that the spectra for nearly all compounds at all pressure conditions are generally well separated, with the most similar spectra being those for ethanol (C_2H_5OH), formic acid (HCOOH), hydrogen sulfide (H_2S), hydrogen cyanide (HCN) and carbonyl sulfide (OCS) corresponding to five mapped points near a location of (5,0) in Fig. 4, all corresponding to spectra calculated at very low pressures. Although these points do not exactly overlap, indicating a degree of separability in the spectra, their similarity suggests the potential for misclassification of these spectra, particularly at very low pressures. The rest of the training data are very well spaced, which should allow ML-based classification.

2.3 ML classifiers

ML approaches to classification can be most simply be represented by a black box as shown in Fig. 5a. Often, data is available from prior experiments or computations. These data can be used as training examples in the supervised ML approaches, in which classifiers learn from labeled training data. If the data is not labeled, an unsupervised ML method can be used to organize similar data into groups [29, 48]. For predicting the compound responsible for an unknown spectrum, ML classifiers needed to be trained with a sufficient number of labeled training spectra. A general supervised ML method is shown in Fig. 5b. The training spectra contain information about the true target function,$\boldsymbol{f}$. The training spectra contains features that are arranged in a feature vector, X. It is also known which features belong to each compound and compound labels are then represented using a label index. These integer values are arranged in a target vector, y. The target function,$\boldsymbol{f}$, represents the true relation that maps X to y. A ML classifier uses a learning algorithm, $\boldsymbol{A}$, and suitable hypothesis set, $\boldsymbol{H}$, to approximate a function, $\boldsymbol{g}$, that approximately captures the target function, $\boldsymbol{f}$. Features of the training spectra are used as examples to determine model parameters and allow the development of a classifier. The performance of a classifier can be adjusted or controlled using hyperparameters, parameters that are unique to each classifier type and influence classifier performance but are not determined from the training data [49]. Once a classifier makes sufficiently accurate predictions on "unseen" spectra, the hypothesis is called a final hypothesis,$\boldsymbol{g}$. This final hypothesis is then used as black box function to predict compound labels from unlabeled measured spectra as shown in Fig. 5a.

In this work, a total of 16 different supervised ML classifiers were built using Scikit-learn machine learning library for Python [50]. The classifiers were developed using two different implementations of eight ML methods. In the regular classification strategy, a single classifier trains from the labeled training spectra for all twelve compounds. In the one-vs-rest (OVR) strategy, a classifier is constructed using multiple internal classifiers where each internal classifier trains for a single compound. All 16 classifiers were then compared for the identification of spectra in the parameter space described above. Classifiers are trained twice, in a 70–30 training–testing split and then again in tenfold cross-validation described in Sects. 2.4.1 and 2.4.2. The classifiers are briefly described below. The hyperparameters discussed in this section are decided in the 70–30 training–testing split and the tenfold cross-validation tests were performed using those same hyperparameters to show how the classifiers perform across the range of pressures.

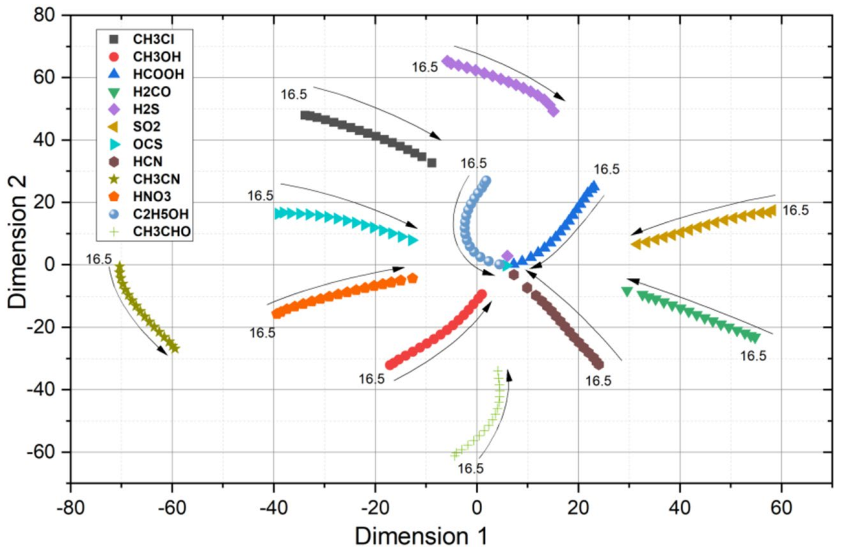


Fig. 4 *t*-Stochastic neighbor embedding (*t*-SNE) of the spectra for the twelve compounds considered. Arrows indicate the direction of decreasing pressure, where the 16.5 label represents the highest pressure data at 16.5 Torr.

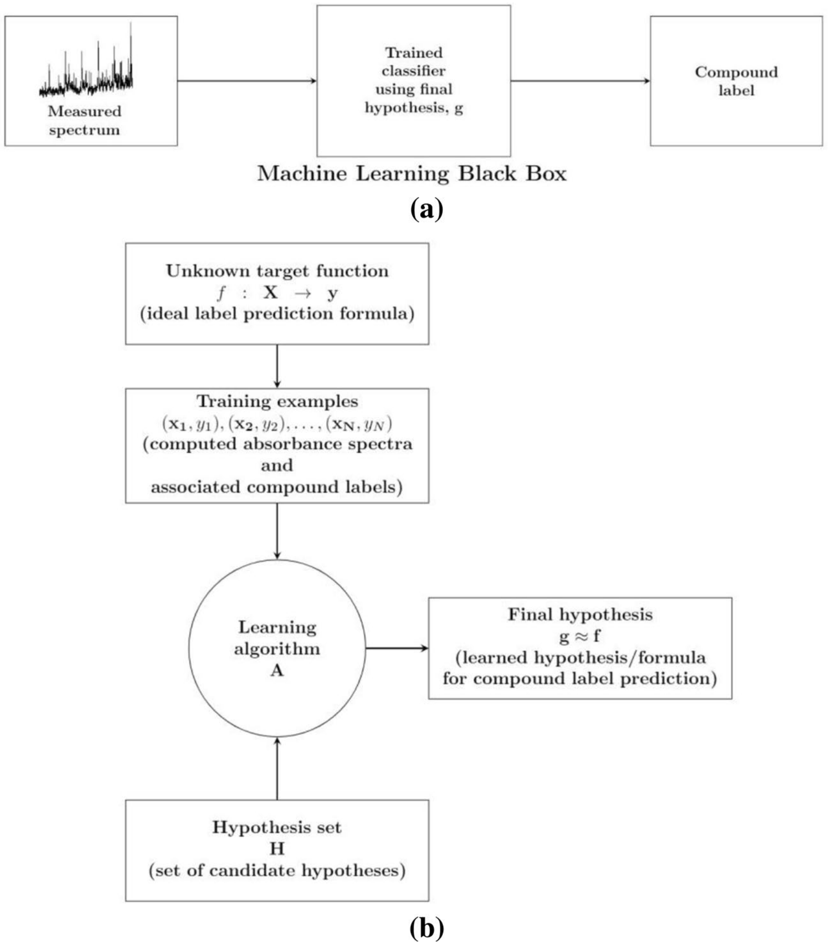

Measured spectrum
Trained classifier using final hypothesis, g
Compound label
Machine Learning Black Box
(a)
Unknown target function $f : \mathbf{X} \rightarrow \mathbf{y}$ (ideal label prediction formula)
Training examples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$ (computed absorbance spectra and associated compound labels)
Learning algorithm A
Final hypothesis $g \approx f$ (learned hypothesis/formula for compound label prediction)
Hypothesis set H (set of candidate hypotheses)
(b)

Fig. 5 **a** Generic ML classifier as a black box for predicting the compound responsible for an unknown spectrum, **b** generic ML approach to the prediction of compound labels. (figure adapted from [48])

K-nearest neighbors (k-NN) k-NN is an instance-based learning algorithm, known for its simplicity, ease of implementation, and low computational expense [51–57]. Within the current work, the k-NN algorithm compares an unknown spectrum to stored training spectra to find the most representative classification. The classification of an unlabeled spectrum is then decided upon based on its similarity with k nearest neighbors, where k is a positive integer and the nearest neighbors are the training spectra which are most similar to the unknown spectrum. The unknown spectrum is classified based on a vote of the k most similar training spectra. To make the k-NN algorithm work, a suitable distance metric with an optimal number of nearest neighbors is required. There are several distance metrics available for use [58] but the optimal distance metric and nearest neighbors are usually found by trial and error [59].

In the present study, a number of nearest neighbors, k was set to three and all neighbors were given uniform weight in voting. The nearest neighbors (most similar training spectra) were chosen using the Euclidean distance metric, given by $D(x, y) = \sqrt[p]{\left|x_i - x'_i\right|^p}$, where x is a feature vector in the training data, x' is a target vector, and, here, p is chosen to be 2. Implementation details of the classifier can be found in the Scikit-learn machine learning library documentation [50].

Decision tree (DT) Over the years, several DT algorithms [60–63] have been developed to solve several real-world problems, such as credit approval, disease diagnosis, and image classification. DT can be an effective tool, particularly when the trees are small; however, DTs have a disadvantage that results in overfitting and often requires "pruning" methods [37]. The growth of a tree is stopped when the training data is sufficiently divided using a stopping criterion or by replacing every subtree with a leaf node when the error rate of the leaf node is lower than the sub-tree. Popular DT algorithms are CART, ID3, C4.5, and C5.0. The Scikit-learn library contains a modified version of the CART algorithm [61, 64] which is used in this work.

The DT algorithm classifies unlabeled data by learning simple decision rules from the training data. These rules are represented in a flow chart or tree structure, where the decision path starts at a root node and proceeds through several branches (internal nodes) to the terminal leaf node which provides the classification. The mathematical formulation of the DT algorithm can be found in Scikit-learn documentation [64]. Briefly, the DT algorithm organizes data by grouping together similar data and dividing the input space of training data. How a tree grows is dictated by the depth of the tree. The depth of the tree refers to the maximum distance between the root node and the farthest leaf node. To group similar data and divide the input space, an impurity metric is used to judge if a leaf node is “impure” and whether it should be further divided. Impure leaf nodes are unable to classify data with 100% classification accuracy. In the present study, the maximum depth of the tree was set to 50 and the ‘entropy’ impurity function, H, was implemented [49, 64]:

$$H(X_m) = -\sum_k p_{mk}\log_b(p_{mk}),$$

where X_m is the training data at node m, b is usually 2, and p_{mk} is the proportion of data at node m with label k. All 229 data points within each spectrum were considered when building a DT classifier based on resampled training spectra and all compounds (classes) were given equal weight.

Random Forests (RF) The RF algorithm fits multiple decision trees to subsets of training data and prevents overfitting by averaging the results of DTs [65, 66]. Each DT is created independently to its full depth and each DT has an equal vote in making the final classification decision, regardless of depth. From the given training data, a number of samples are randomly chosen in a process known as bootstrapping. DTs are created based on the selection of an optimal number of features from the bootstrapped datasets. All the DTs in the forest then make a classification decision, a step known as bagging. The absent training data in the bootstrapped dataset is called the out-of-bag dataset. The out-of-bag samples are used for testing the random forest classifier. The fraction of the out-of-bag data that are incorrectly classified is called the out-of-bag error (OOBE). Based on the OOBE, an optimal number of DTs is chosen for RF classification performance. In the present study, we used five DTs to build the forest, with a maximum depth of five-leaf nodes in each tree. The entropy function was used as a measure of impurity at leaf nodes.

Support Vector Machine (SVM) Support vector machines use a hyperplane to construct a decision surface and performs spatial separation of data to achieve the widest possible gaps between different classes [67] and, hence, the optimal decision surface. The hyperplane is determined from some of the training data points, the support vectors, providing margin or distance between the hyperplane and those support vectors. If the data is not linearly separable, a SVM classifier will allow some misclassifications. For such linearly non-separable cases, the margin is called a “soft margin” and the classifier is a “soft margin” classifier [28]. SVM was originally conceived as a binary classifier and performed exceptionally well in classifying text and images. However, SVM can also handle multiclass classification [68] and regression [69] problems and is more immune to overfitting compared to other ML methods [70].

SVM classifiers can perform exceptionally well on linearly non-separable data using the kernel trick [71]. Using a kernel function, the classifier can transform the linearly non-separable data to a higher dimensional space where a hyperplane will be able to separate the data. However, to make the computations faster, SVM simply calculates the relationship between pairs of data points if they were transformed to a higher dimensional space. This process of computing the relationship between the training data points without actually transforming them is known as the kernel trick [23]. For the implementation details of a SVM classifier, see [11, 49, 50].

In the present work, two different SVM classifiers were constructed. One with a linear kernel function (SVM-linear) and the other with a radial basis function kernel (SVM-RBF). We used LIBSVM library for support vector classification in Scikit-learn library to construct the classifiers [50]. The RBF kernel is generally advantageous because training spectra are mapped nonlinearly to an infinite-dimensional space to allow SVMs to tackle nonlinear relations between features and compound label index. Choosing between the linear and RBF kernel is a challenging task and can be achieved by cross-validation. To make the choice, it is recommended to completely search the critical hyperparameter space of both kernels. The RBF kernel-based SVM has two hyperparameters, namely, the soft margin constant (C value) and the kernel coefficient (gamma value). However, the linear kernel-based SVM does not require any kernel coefficients and can be constructed using only the soft margin constant value. For training spectra with a large number of features, mapping to a higher dimension may not be required since it may not result in any performance improvement and a linear kernel-based SVM may be a simpler and a better choice [72]. Furthermore, the linear kernel has been shown to be a special case of RBF kernel [73].

The value of the soft margin constant (C value) was set to 100 and the kernel coefficient (gamma value) was set to 0.125 for SVM-RBF classifier and its OVR implementation. The SVM-linear classifier used $C=260$ and the OVR(SVM-linear) classifier was constructed with $C=1$.

Multi-layer Perceptron (MLP) A multi-layer perceptron (MLP) is a special class of feedforward artificial neural networks with a backpropagation algorithm. MLP is composed of a number of nodes arranged in at least three layers. In general, there are one or more hidden layers sandwiched between an input layer and an output layer. A layer consists of computation nodes, which are called neurons. Each neuron can be activated or “fired” using an activation function [74] which is differentiable and non-decreasing [75].

The artificial neurons mimic neurons in the brain and process signals and communicate with each other. However, instead of using electrical and chemical processes like in human neurons, the artificial neurons use weighted and biased combinations of inputs and outputs for communication. The neurons learn complex relations from the training data which are given as inputs in the input layer. In supervised learning, the output layer consists of a single neuron. Each neuron in the input layer then learns from a single feature of the training data [29]. The neuron in the output layer receives the target for each set of features. The hidden layers connect the input and output layers to train the network. Typically, there are two phases to train a MLP. In the forward phase, the inputs (features), network parameters (weights and biases), and the desired output of the model are set. The model parameters are initiated randomly. The input signal from the input layer of neurons passes through each layer and an error signal is computed using an activation function. In the backward phase, the error signal proceeds from the output layer towards the input layer with appropriate updates made to model parameters using backpropagation algorithms [29, 74, 76, 77]. For mathematical details regarding the implementation of MLP, readers are referred to [49, 50].

In the present study, the multi-layer perceptron receives training spectra in the form of 229 features; i.e., the input layer is comprised of 229 neurons. We used an identity function as the activation function. There are two hidden layers, with five and six neurons in each, as shown in Fig. 6. The weights are randomly initialized and are updated to progressively minimize the cross-entropy loss function. In an iteration, first the loss is calculated in the forward phase and then a backpropagation with stochastic gradient descent (SGD) is performed. The backpropagation starts from the output layer and proceeds to previous layers and updates each weight parameter, thus decreasing the loss function.

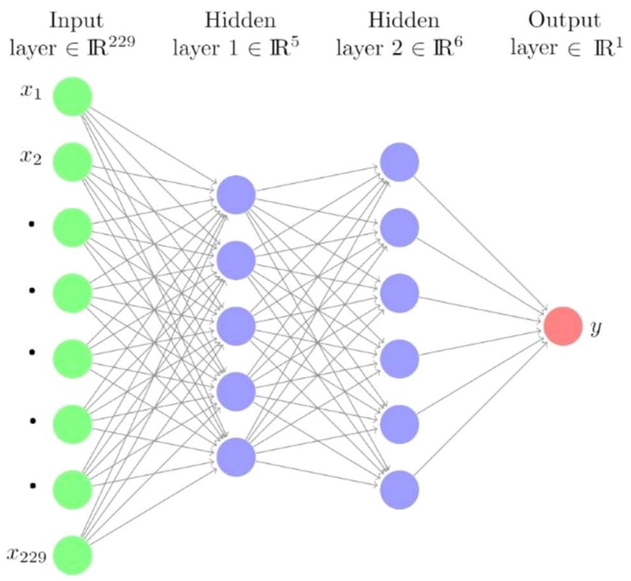

Input layer $\in \mathbb{R}^{229}$
Hidden layer 1 $\in \mathbb{R}^5$
Hidden layer 2 $\in \mathbb{R}^6$
Output layer $\in \mathbb{R}^1$
x_1
x_2
x_{229}
y

Fig. 6 Multi-layer Perceptron (MLP) with two hidden layers. x refers to individual features and y refers to label integer index value

Adaptive Boosting Classifiers The Adaptive Boosting algorithm is based on a boosting technique that combines the performance of multiple weak learning methods (estimators) to produce a superior method [78–80]. In this work, we created two Adaptive Boosted classifiers. One uses DTs as estimators and combines their outputs to classify spectra. The other one uses RFs as estimators and combines the output of each forest.

Adaptive Boosted Decision Trees (ABDT) ABDT ensembles the outputs of decision trees (DT) to boost their performance. Therefore, this classifier uses decision trees as weak learners. Decision trees are created sequentially as a forest of trees. Initially, weights are applied to each spectrum in the training data. The first tree is trained on the entire training spectra and makes classifications. The next tree corrects for the misclassifications of the first tree by adjusting the weights. This process is carried out by all successive trees to build the classifier. This process is fundamentally different from the majority voting by independent decision trees inside a conventional random forest algorithm. Furthermore, ABDT significantly improves the learning performance of a single decision tree, since the successive trees purposefully improve the misclassifications. In our construction of the ABDT classifier, we used ten decision trees, each with the same hyperparameters as the regular DT classifier.

Adaptive Boosted Random Forest (ABRF) In this instance of a boosted classifier, we use random forests as weak learners to construct the classifier. Therefore, these weak learners (RFs) are built sequentially [80], with each newly built forest attempting to improve the performance of the previously built forest by adjusting the weights applied to each spectrum in the training data. This learning process is different from a single random forest, where the bagging technique is employed. In the bagging technique, the learners (decision trees inside a single forest) are built-in parallel and independent of each other [78] and the final output of the classifier is based on majority voting by the decision trees inside the forest. We employed 20 random forests to construct the ABRF classifier. Each random forest consisted of ten decision trees with a maximum depth of five leaf nodes.

One-Vs-Rest (OVR) implementation of the above eight classifiers The OVR method is a binary classification strategy in which a single classifier is trained to recognize each class or compound [81]. In this implementation strategy, the goal of the classifier is to take a base classifier and fit one classifier per class. For example, an OVR(SVM-linear) classifier uses a base SVM-linear classifier and creates twelve different incarnations of binary SVM-linear classifiers to individually train on each class/compound. Each incarnation of the base classifier trains for a single compound against the eleven other compounds treated as one compound. For example, if the base classifier is trained for chloromethane, it simply treats all training spectra from the other 11 compounds as "not-chloromethane" and learns how to identify chloromethane apart from the other compounds. Thus, the method simply consists of fitting one base classifier per compound. When OVR(SVM-linear) is called upon, each internal incarnation of the SVM-linear classifier then evaluates the unknown spectrum and tries to match it with its compound class label. The hyperparameters used for OVR implementation of each classifier are not necessarily same as that of the regular implementations of the classifier.

2.4 ML implementation

The sixteen ML classifiers are tested for their ability to identify a pure component spectrum in the 220–330 GHz region. The classifiers are those described above (k-NN, DT, RF, SVM-linear, SVM-RBF, MLP, ABDT, and ABRF, and OVR implementation of those eight ML methods). A Python 3 computer program supported by the Scikit-learn machine learning library [50] was written to implement all the classifiers. The program was executed on a Core i7 laptop with a CPU clock speed of 1.8 GHz and having 16 gigabytes of random access memory. The training data required 36 megabytes of disk space.

The absorption spectrum classification program was implemented according to the flowchart structure given in Figs. 7 and 8. The program is initiated by passing input variables: pressure and pressure increment, temperature, frequency and frequency increment, optical path length, and the database of spectroscopic parameters for the chemical compounds of interest. These inputs are passed to a modified HAPI interface where spectral simulations are carried out. The number of compounds and the number of training spectra per compound must also be specified to generate a training dataset of simulated spectra. In the

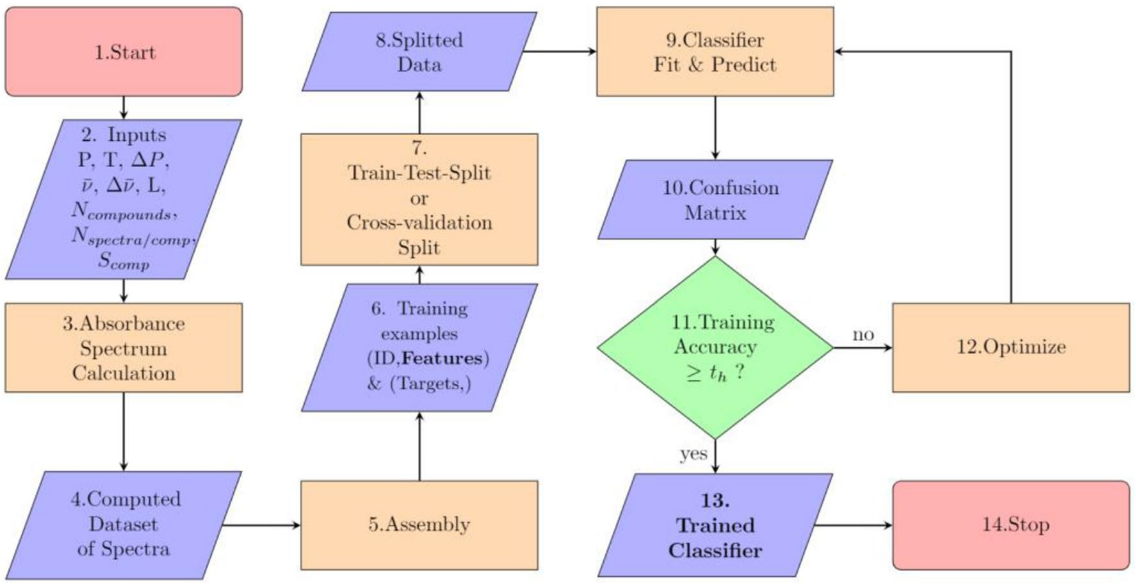

1.Start
2. Inputs P, T, ΔP, $\bar{\nu}$, $\Delta\bar{\nu}$, L, $N_{compounds}$, $N_{spectra/comp}$, S_{comp}
3.Absorbance Spectrum Calculation
4.Computed Dataset of Spectra
5.Assembly
6. Training examples (ID,**Features**) & (Targets,)
7. Train-Test-Split or Cross-validation Split
8.Splitted Data
9.Classifier Fit & Predict
10.Confusion Matrix
11.Training Accuracy $\geq t_h$?
no
yes
12.Optimize
13. Trained Classifier
14.Stop

Fig. 7 ML classifier data generation and training procedure. For cross-validation, the procedure is stopped after step 10 and repeated ten times for ten different folds. Steps 1–5 illustrate the generation and assembly of simulated spectra and not actually part of the training procedure

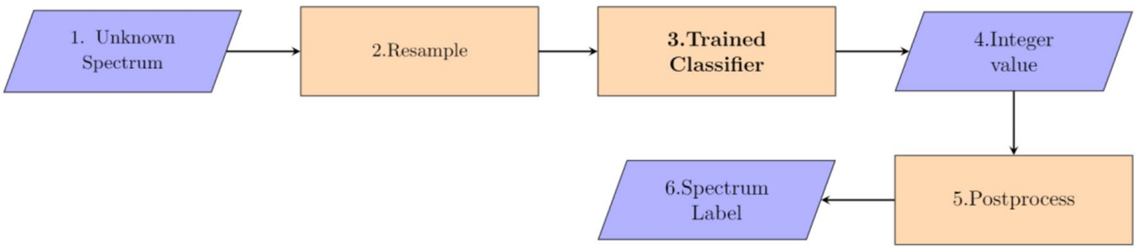


Fig. 8 ML classifier testing and validation procedure. If a simulated spectrum is tested, step 2 is skipped, since the spectrum does not need resampling

present study, twelve compounds and 164 computed spectra per compound were chosen for the training dataset.

The calculated absorption spectra are assembled into two numpy arrays. One array, $\boldsymbol{X}$, consists of an index and the features of each spectrum (absorbance value for each wavenumber sample). Another array, denoted by $\boldsymbol{y}$, consists of the labels/classification for each spectrum and is called the target array. All the simulated absorption spectra can be used as training examples; however, we randomly remove some of the spectra to form a testing data set. During the formation of the training dataset and the testing dataset, spectra are selected equally for each compound to maintain stratification of the training set. In this study, classifiers are trained twice, in a 70–30 training–testing split and a tenfold cross-validation, as described below.

2.4.1 70–30 training–testing split

As the first training step, 70% of the computed absorption spectra are randomly selected as the training dataset and the remaining 30% is held for testing and optimizing the performance of the trained classifiers. Training and testing spectra are selected by stratification of the classes, therefore, each compound has approximately the same number of training and testing spectra. To assess performance, classifiers fit 1357 training spectra to learn features of the data.

Once trained, the ML classifiers are tested for their classification accuracy against the 611 testing spectra. The testing process is shown in Fig. 8. Each testing spectra is treated as an unknown spectrum. Since the testing spectra are also simulated and have the same number of features as the training spectra, no resampling is required. The trained classifier receives each testing spectrum and outputs an integer label index ranging from 0 to 11, revealing the compound associated with the spectrum. Afterwards, classification accuracy and other metrics are computed from the confusion matrices of each classifier. We set an accuracy threshold (t_h) of 99%. If classification accuracy is greater than 99%, performance is deemed satisfactory for the prediction of simulated spectra. Accuracy is further interrogated by inspecting the confusion matrices of the classifiers and examining different classification metrics (the F1 score, precision, and recall). If the classification accuracy for the testing data is below 99%, the hyperparameters of the ML classifier are adjusted and the classifier is retrained. Once all the classifiers meet the 99% threshold, we proceed to use these classifiers for a stratified tenfold cross-validation to further analyze the performance of the classifiers across the training examples and to determine if the size and quality of training examples are sufficient. The performance of the 70–30 split trained classifiers are averaged over three iterations, to account for randomness, and are given in Table 2. As shown in Table 2, all classifiers met the accuracy threshold of 99% and most

Table 2 Performance of ML classifiers in the 70–30 training–testing split studies (1357 training and 611 testing spectra)

Classifier	Runtime [ms]	Accuracy [%]
k-NN	195 ± 38	99.89 ± 0.09
DT	389 ± 27	99.29 ± 0.34
RF	130 ± 2	99.02 ± 0.17
SVM-linear	255 ± 17	100.0 ± 0.00
SVM-RBF	462 ± 52	99.46 ± 0.41
MLP	257 ± 32	99.89 ± 0.09
ABDT	3191 ± 430	99.95 ± 0.09
ABRF	2194 ± 68	99.84 ± 0.28
OVR(k-NN)	784 ± 105	99.89 ± 0.09
OVR(DT)	1011 ± 106	99.40 ± 0.25
OVR(RF)	278 ± 49	99.29 ± 0.25
OVR(SVM-linear)	568 ± 135	99.95 ± 0.09
OVR(SVM-RBF)	462 ± 40	99.95 ± 0.09
OVR(MLP)	456 ± 123	99.95 ± 0.09
OVR(ABDT)	1016 ± 145	99.45 ± 0.25
OVR(ABRF)	893 ± 112	99.67 ± 0.00

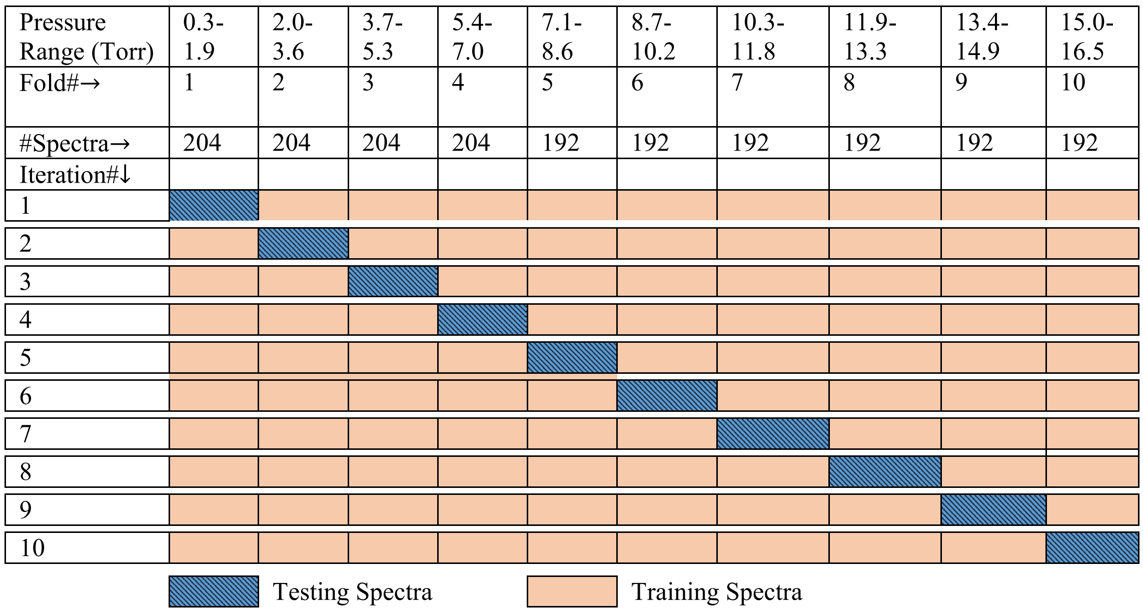
Values are averaged over three random trials where training–testing spectra were selected using Python random number generator seeds

classifiers require less than 1 s of computer clock time to train. SVM-linear classifier is the best performing classifier on the simulated spectra.

2.4.2 Stratified tenfold cross-validation

We performed a tenfold cross-validation, as illustrated in Fig. 9, to assess the performance and stability of classifiers, the quality of the training examples, and if the number of the training examples is sufficient. Ten-fold cross-validation is equivalent to separating the training examples according to a 90–10% training–testing split. The training dataset is divided into ten separate folds according to their pressure. Nine folds are used for training and one of the folds is used for testing. Each class of compound is equally represented in each fold; hence, our implementation of the process is known as stratified tenfold cross-validation.

To perform the tenfold cross-validation, we train classifiers ten times over ten different iterations. We used the same hyperparameters that resulted from the 70 to 30 optimization of the classifiers. In each iteration, a different fold was used for testing and the other nine folds for training. From Fig. 9, we observe that in the first iteration we use spectra belonging to folds 2 through 10 for training, which were simulated for pressures from 2.0 to 16.5 Torr. Spectra in fold 1 are used for testing, which were simulated at pressures of 0.3–1.9 Torr. As we progress through the iteration, a complete sweep over the entire training space is made, and classification performance across different pressure ranges



Pressure Range (Torr)	0.3-1.9	2.0-3.6	3.7-5.3	5.4-7.0	7.1-8.6	8.7-10.2	10.3-11.8	11.9-13.3	13.4-14.9	15.0-16.5
Fold#→	1	2	3	4	5	6	7	8	9	10
#Spectra→	204	204	204	204	192	192	192	192	192	192
Iteration#↓										
1	▨	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
2	🟧	▨	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
3	🟧	🟧	▨	🟧	🟧	🟧	🟧	🟧	🟧	🟧
4	🟧	🟧	🟧	▨	🟧	🟧	🟧	🟧	🟧	🟧
5	🟧	🟧	🟧	🟧	▨	🟧	🟧	🟧	🟧	🟧
6	🟧	🟧	🟧	🟧	🟧	▨	🟧	🟧	🟧	🟧
7	🟧	🟧	🟧	🟧	🟧	🟧	▨	🟧	🟧	🟧
8	🟧	🟧	🟧	🟧	🟧	🟧	🟧	▨	🟧	🟧
9	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	▨	🟧
10	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	▨

▨ Testing Spectra 🟧 Training Spectra

Fig. 9 Schematic representation of tenfold cross-validation. The pressure range refers to the pressures of the spectra belonging in each fold. In each iteration, classifier accuracy, precision, recall, and F1 scores are determined. Folds 1–4 contains 204 spectra each and folds 5–10 contains 192 spectra each

is evaluated. Cross-validation also provides information as to whether the training data is of sufficient size and if there are any potential weakness within the training set [82] and reveals how sensitive classifiers are to particular parameters. For example, we can observe that if we only use classifiers trained at higher pressures, they fail to correctly identify spectra at lower pressures. On the other hand, if we train classifiers at lower pressures they can recognize higher pressure spectra, as illustrated in cross-validation results.

Stratified tenfold cross-validation showed that all classifiers trained via the 70–30 training–testing split were satisfactory; hence, we tested these classifiers against experimental spectra. The classifiers trained using the 70–30 training–testing split were used, instead of those that resulted following tenfold cross-validation (90–10 training–testing split), because it has been shown that ML classifiers that have been suboptimally trained on simulated data can perform better than those that have been over-trained [83].

All 16 classifiers were tested against experimental measurements using the procedure shown in Fig. 8. The measurements [7] were made at much higher spectral resolution than the ML classifiers were trained on; therefore, all experimental spectra were first resampled using a Fast Fourier Transform to generate spectra with resolution matching the simulations (0.016 cm^{-1}). A resampled unknown experimental spectrum is received by the trained ML classifier function as an input. The trained classifier then processes the spectrum and produces a label ranging from 0 to 11 corresponding with the assigned chemical compound for that spectrum.

3 Results and discussion

3.1 Tenfold cross-validation

The performance of ML classifiers for classifying unknown spectra was first assessed here by confusion matrices. Confusion matrices present the number of instances each compound is correctly and incorrectly identified and enable easy determination of classification performance metrics. Performance of the ML classifiers is better understood in terms of four statistical parameters:

a. Classification accuracy (CA): The percentage of correctly classified spectra of all tested spectra.
b. Precision (P): Precision is defined as the ratio of true positives (T_P, correctly predicts the positive class/compound) over the sum of true positives and false positives (F_P, incorrectly predicts the positive class). It is a measure of a ML classifier's ability to not misidentify a given spectrum and answers the question of what fraction of spectra identified as positives were truly positive.

$$P = \frac{T_P}{T_P + F_P}.$$

c Recall (R): Recall is the ability of a ML classifier to find all positive samples of a given compound and is defined as the ratio of true positives over the sum of true positives and false negatives (F_N, incorrect predictions of the negative class). Recall indicates the fraction of actual spectra for a particular compound that are correctly predicted.

$$R = \frac{T_P}{T_P + F_N}.$$

d. Score (F1): Score or F1 score is the harmonic mean of precision and recall.

$$F1 = 2 \times \frac{P \times R}{P + R}.$$

Figure 10 illustrates the confusion matrices for tenfold cross-validation for four classifiers for three different iterations of the tenfold cross-validation: iteration 1, containing testing spectra at the lowest pressures of 0.3–1.9 Torr and training spectra at pressures of 2.0–16.5 Torr; an inner iteration falling between iterations 2–9; and iteration 10, containing testing spectra at the pressure of 15.0–16.5 Torr and training spectra at pressures of 0.3–14.9 Torr. For all confusion matrices (all compounds, all classifiers, all iterations), please see the appended supplementary material.

The confusion matrices illustrate the predictions of various ML classifiers for testing spectra, where each row shows the molecule for the known testing spectra and each column shows the molecule predicted by the classifier. In each iteration, either 16 or 17 testing spectra were considered for each molecule. The numbers along the diagonal of the confusion matrices indicate correct predictions by the classifiers and off-diagonal numbers indicate incorrect predictions. The color scheme simply distinguishes between the ML methods used to construct the classifiers.

From Fig. 10 it is observed that the classifiers perform relatively worse on iteration 1 (lowest pressures), indicating that when ML classifiers are trained on higher pressure spectra, they are prone to misclassify when asked to extrapolate to lower pressure, where spectral features are much weaker and difficult to distinguish. Fortunately, in most cases at these low pressures, the features that are so weak that they are not experimentally resolvable, making this poorer performance an often unimportant result.

For iterations, 2–9, all ML classifiers, except DT, OVR(DT), and OVR(ABDT), have a performance which can be characterized as producing negligible misclassifications. This is expected because for the inner folds the classifiers are trained on both lower pressure and higher pressure spectra and hence the classifiers are not asked to extrapolate beyond the training dataset. Decision trees are generally more prone to "overfitting". Overfitting refers to memorization of peculiarities of training data instead of learning predictive rules [83]. A consequence of overfitting is that a classifier may perform very well in the prediction from training data but will fit noise in any new data, leading to misclassification. Decision tree methods misclassify spectra throughout all the iterations, both in regular and OVR implementations. Therefore, it is likely that DTs are alone insufficient for finding general classification rules. This observation is further supported by the fact that the RF classifier and boosted decision trees (ABDT) classifier both generally perform better than the DT classifier.

For the 10th iteration, some misclassifications are also observed, in this case for both the DT and RF-based classifiers: DT, OVR(DT), OVR(ABDT), RF, OVR(RF), and OVR(ABRF). All other ML classifiers do very well on the 10th iteration and for iterations 2–9, in terms of classifying all twelve compounds. The misclassifications, again, likely arise from extrapolation; i.e., training on low-pressure spectra and testing on a high-pressure spectrum. With increasing pressure, relatively weaker peaks become more prominent; hence, if ML classifiers are trained on low-pressure spectra, they do not learn as much from these weak peaks that then appear more prominently in the high-pressure testing data. The compounds for which the classification performance are worst are those that contain the weakest peaks and include formic acid, formaldehyde, and sulfur dioxide. Another important observation is while at iteration 1, all ML classifiers misclassify, however, at iteration 10, only the tree-based methods misclassify. This indicates that tree-based methods are highly likely to overfit the training spectra generated at lower pressures.

Table 3 presents a comparison of the combined computer clock time for training and testing of each ML classifier in tenfold cross-validation. It is observed that the OVR implementations are generally faster in training and testing. An exception is the OVR(DT) classifier, which took longer to train compared to the DT classifier. The slowest training times are generally observed for adaptive boosted tree-based classifiers, namely ABDT and ABRF classifiers and their OVR counterparts, due to the sequential nature of the boosting method.

Table 3 also presents a comparison of average values and standard deviations of accuracy, precision, recall, and F1 score for the classifiers in tenfold cross-validation. The tree-based methods (DT and RF, both in regular, boosted, and OVR implementation) have the worst performance, in terms of all classification metrics, a result in keeping with the confusion matrices. All other classifiers have an accuracy of greater than 95% with the OVR(SVM-linear) and OVR(SVM-RBF) classifiers at greater than 98% accuracy and with the smallest standard deviations in accuracy. The k-NN and MLP classifiers perform similarly to the SVM classifiers; however, the performance improvement of OVR(k-NN) and OVR(MLP) in comparison to k-NN and MLP is negligible, while the OVR(SVM) classifiers perform better than the standard

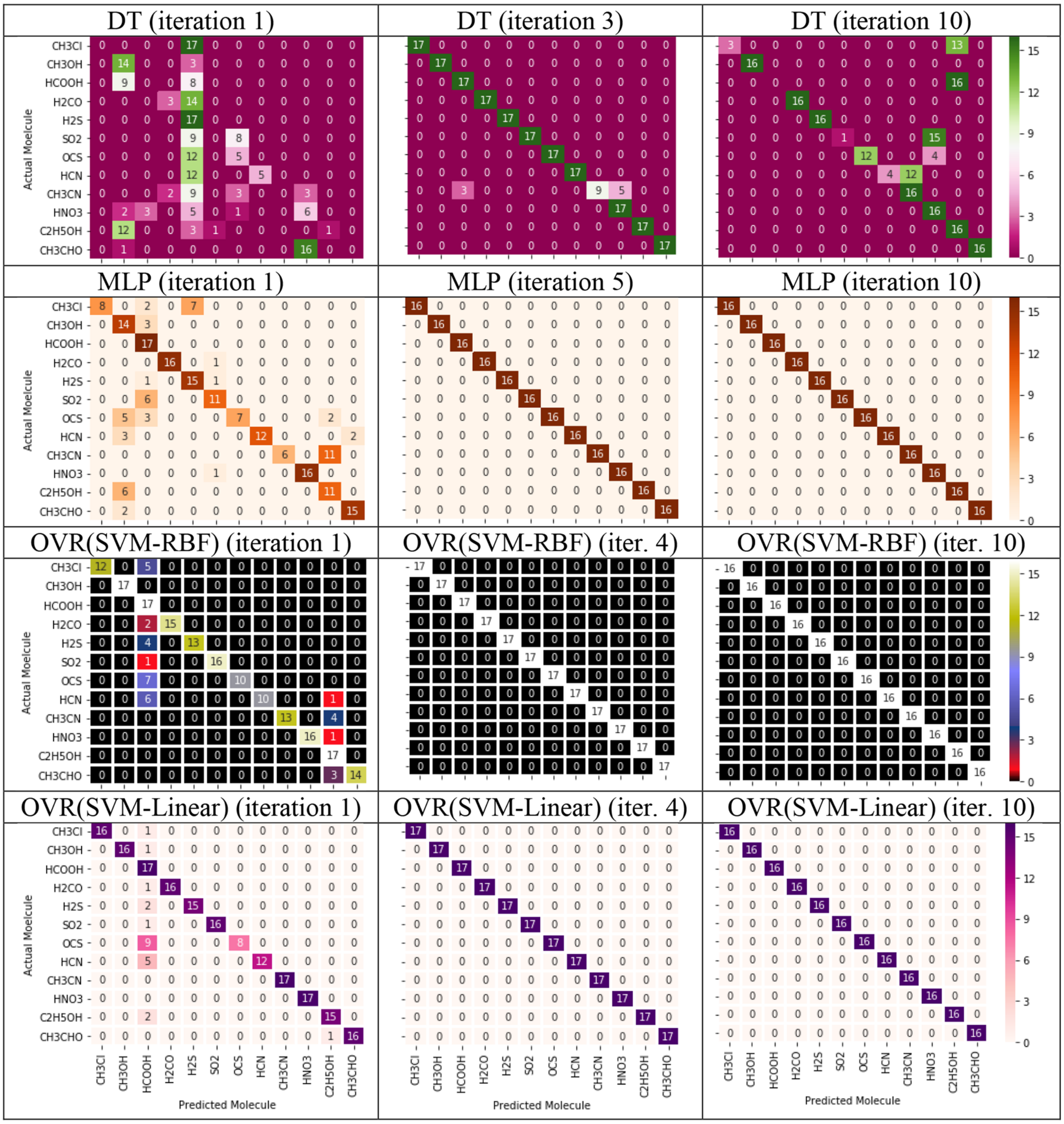


Fig. 10 Confusion matrices, represented as heat maps, for cross-validation results of DT, MLP, OVR(SVM-RBF), and OVR(SVM-linear) classifiers

SVM classifiers for the classification of simulated data. Overall, the best performing methods are SVM-linear, SVM-RBF, MLP, and k-NN in both regular and OVR implementations, with average classification accuracy above 96% overall ten folds. Precision for these highest-performing classifiers is 97–99%, indicating they are very unlikely to yield false positives. Recall and F1 scores also show strong performance at greater than 97% for these methods (SVM-linear, SVM-RBF, MLP, and k-NN in both regular and OVR implementation).

Table 3 Performance for ML classifiers in tenfold cross-validation studies on simulated spectra. Values reported here are averaged over the 10 iterations

Classifier	Runtime [ms]	Accuracy [%]	Precision	Recall	F1 Score
k-NN	931 ± 123	97.0 ± 9.5	0.99 ± 0.04	0.97 ± 0.1	0.97 ± 0.1
DT	1181 ± 127	88.3 ± 24.3	0.90 ± 0.2	0.88 ± 0.2	0.87 ± 0.3
RF	886 ± 115	92.4 ± 19.2	0.95 ± 0.1	0.92 ± 0.2	0.93 ± 0.2
SVM-linear	235 ± 52	97.2 ± 8.84	0.99 ± 0.04	0.97 ± 0.1	0.97 ± 0.1
SVM-RBF	667 ± 75	97.1 ± 9.3	0.99 ± 0.04	0.97 ± 0.1	0.97 ± 0.1
MLP	1025 ± 99	97.3 ± 8.7	0.98 ± 0.1	0.97 ± 0.1	0.97 ± 0.1
ABDT	4025 ± 1119	93.0 ± 22.2	0.95 ± 0.2	0.93 ± 0.2	0.93 ± 0.2
ABRF	2484 ± 1207	93.9 ± 19.4	0.97 ± 0.1	0.94 ± 0.2	0.94 ± 0.2
OVR(k-NN)	599 ± 192	97.0 ± 9.5	0.99 ± 0.04	0.97 ± 0.1	0.97 ± 0.1
OVR(DT)	1323 ± 178	88.7 ± 23.0	0.91 ± 0.2	0.89 ± 0.2	0.88 ± 0.2
OVR(RF)	289 ± 23.0	89.7 ± 23.9	0.92 ± 0.2	0.90 ± 0.2	0.89 ± 0.3
OVR(SVM-linear)	628 ± 69	98.9 ± 3.6	1.0 ± 0.02	0.99 ± 0.03	0.99 ± 0.03
OVR(SVM-RBF)	605 ± 117	98.3 ± 5.3	0.99 ± 0.02	0.98 ± 0.1	0.99 ± 0.1
OVR(MLP)	560 ± 145	96.5 ± 11.0	0.99 ± 0.02	0.97 ± 0.1	0.97 ± 0.1
OVR(ABRF)	995 ± 197	92.2 ± 21.9	0.95 ± 0.1	0.92 ± 0.2	0.92 ± 0.2
OVR(ABDT)	1202 ± 2	87.4 ± 22.9	0.87 ± 0.2	0.87 ± 0.2	0.86 ± 0.2

The fractions of training spectra and testing spectra in each iteration were not necessarily same

3.2 Performance against experimental spectra (classifier validation)

The performance of the ML classifiers was tested in the classification of 36 experimental spectra, measured in our laboratory. Since we randomly trained and tested ML classifiers with a 70–30 training–testing split three different times, we have three different confusion matrices for validation of the ML classifiers with experimental spectra. One of the confusion matrices from these three random trials is presented in Table 4, the rest are given in supplementary material. The first column in Table 4 lists the names of the compounds associated with each experimental spectra, where the hyphenated number refers to experimental pressure in Torr. The experimental data set contains both filtered and unfiltered measurement to test the extent of overfitting of the classifiers. With the exception of acetonitrile (CH_3CN), all compounds have at least one repeated measurement included in the validation set, allowing a test of the robustness of the classifiers to slight variations in experimental spectra. From Table 4, it is evident the MLP, OVR(SVM-RBF), and OVR(SVM-linear) classifiers outperform the rest of the classifiers in terms of recognition of the experimental spectra. Interestingly, the OVR(MLP) and SVM-RBF and SVM-linear classifiers do not perform nearly as well.

As expected, based on the cross-validation results, DT and RF classifiers perform very poorly on experimental data, irrespective of regular or OVR implementation. Performance improvement due to boosting is also evident, where ABRF and ABDT classifiers perform better than DT and RF classifiers. When implemented in OVR strategy, ABRF and ABDT performance is improved further. So boosting and OVR combined can be a good approach to implement tree-based ensemble methods to recognize spectra; although the training time for these classifiers may be longer than non-tree-based classifiers. However, it should be noted that classifier performance is governed by the ML method itself but also by the limitations of the training dataset.

Average performance metrics for the classifiers against experiments are given in Table 5. It is clear that for the classification of these 36 experimental spectra, the MLP, OVR(SVM-RBF), and OVR(SVM-linear) classifiers performed best, with greater than 85% average classification accuracy. Average classification accuracy of k-NN, ABDT, and ABRF and their OVR counterparts lies between 75 and 81% and these classifiers can be deemed as good performers. In terms of standard deviation in classification accuracy, precision, recall and F1 score, OVR(SVM-linear) outperforms all the other classifiers. A comparison of the classifiers indicates that the MLP and OVR(MLP) classifier struggles slightly with the identification of methanol, acetaldehyde, and formic acid, which have somewhat similar spectra. OVR(SVM-linear) has difficulty with acetonitrile. The average F1 score for OVR(SVM-linear) is comfortably greater than that for MLP. Therefore, OVR(SVM-linear) may be a better practical choice for the classification of experimental spectra, even with its slightly lower average recall value for some compounds. Also of note, while the regular implementations of the SVM-linear and SVM-RBF classifier have a quite good F1 score for some compounds, both have very poor classification accuracy. Furthermore, the MLP method misclassified three spectra for three different compounds (ethanol, formic acid, and acetaldehyde), while the OVR(SVM-linear) method misclassified three

Table 4 Confusion matrix for the classification of 36 unknown experimental spectra

Classifier type	Regular classifiers								One-vs-Rest (OVR) classifiers							
Experimental spectrum	k-NN	DT	RF	SVM -linear	SVM -RBF	MLP	ABDT	ABRF	k-NN	DT	RF	SVM -linear	SVM -RBF	MLP	ABDT	ABRF
C_2H_5OH-16	✓	CH_3CHO	✓	✓	✓	✓	✓	✓	✓	SO_2	✓	✓	✓	✓	✓	✓
C_2H_5OH-8	✓	CH_3OH	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C_2H_5OH-2	✓	CH_3OH	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C_2H_5OH-4	✓	CH_3OH	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C_2H5OH-8	✓	CH_3OH	✓	✓	✓	✓	✓	✓	✓	CH_3CHO	✓	✓	✓	✓	✓	✓
C_2H_5OH-1	✓	CH_3OH	✓	✓	✓	✓	✓	✓	✓	CH_3CHO	✓	✓	✓	HCOOH	✓	✓
HCOOH-1	OCS	OCS	H_2S	OCS	✓	✓	✓	H_2S	OCS	CH_3CHO	H_2S	✓	✓	OCS	H_2S	H_2S
HCOOH-16	✓	HNO_3	✓	✓	✓	✓	✓	H_2CO	✓	CH_3CHO	✓	✓	✓	✓	✓	✓
HCOOH-4	✓	H_2CO	H_2S	✓	✓	✓	✓	✓	✓	CH_3CHO	H_2S	✓	✓	✓	✓	✓
HCOOH-1	OCS	OCS	HCN	OCS	OCS	CH_3OH	OCS	✓	OCS	CH_3CHO	C_2H_5OH	✓	✓	✓	✓	✓
HCOOH-2	✓	HNO_3	C_2H_5OH	OCS	✓	✓	✓	✓	✓	CH_3CHO	✓	✓	✓	✓	✓	✓
HCOOH-4	✓	HNO_3	✓	✓	✓	✓	✓	✓	✓	CH_3CHO	✓	✓	✓	✓	✓	✓
CH_3OH-1	OCS	✓	HNO_3	OCS	HCOOH	✓	✓	C_2H_5OH	OCS	CH_3CHO	C_2H_5OH	✓	✓	✓	CH_3CN	CH_3CN
CH_3OH-2	✓	✓	HNO_3	C_2H_5OH	✓	✓	✓	✓	✓	CH_3CHO	C_2H_5OH	✓	✓	✓	✓	✓
CH_3OH-8	✓	✓	✓	C_2H_5OH	✓	✓	✓	✓	✓	CH_3CHO	HNO_3	✓	✓	✓	✓	✓
CH_3OH-1	OCS	✓	✓	OCS	OCS	✓	HNO_3	✓	OCS	CH_3CHO	C_2H_5OH	✓	✓	✓	HCOOH	HCOOH
CH_3OH-2	✓	✓	HNO_3	C_2H_5OH	C_2H_5OH	✓	✓	✓	✓	CH_3CHO	C_2H_5OH	✓	✓	✓	✓	✓
CH_3OH-4	✓	✓	✓	C_2H_5OH	C_2H_5OH	✓	✓	✓	✓	CH_3CHO	HNO_3	✓	✓	✓	✓	✓
CH_3Cl-0.5	✓	CH_3CN	HNO_3	✓	✓	✓	SO_2	SO_2	HNO_3	HCN	C_2H_5OH	✓	✓	✓	HCN	HCN
CH_3Cl-1	✓	CH_3CN	HNO_3	✓	✓	✓	✓	C_2H_5OH	✓	✓	CH3CN	✓	✓	✓	✓	✓
CH_3Cl-10	✓	CH_3OH	C_2H_5OH	✓	✓	✓	✓	✓	✓	✓	HCN	✓	✓	✓	✓	✓
CH_3Cl-5	✓	CH_3CN	C_2H_5OH	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CH_3Cl-1	✓	CH_3CN	C_2H_5OH	HNO_3	HNO_3	✓✓	SO_2	✓	✓	H_2S	C_2H_5OH	✓	HNO_3	✓	SO_2	SO_2
CH_3Cl-8	✓	CH_3CHO	CH_3OH	✓	✓	✓	✓	SO_2	✓	H_2S	HCN	✓	✓	✓	✓	✓
CH_3CN-0.5	C_2H_5OH	HCN	CH_3Cl	C_2H_5OH	C_2H_5OH	✓	HCOOH	✓	C_2H_5OH	CH_3CHO	✓	✓	C_2H_5OH	C_2H_5OH	✓	✓
CH_3CN-1	✓	✓	CH_3Cl	✓	✓	✓	✓	✓	✓	C_2H_5OH	C_2H_5OH	✓	✓	C_2H_5OH	✓	✓
CH_3CN-16	✓	SO_2	✓	✓	✓	✓	CH_3CHO	✓	✓	CH_3CHO	✓	C_2H_5OH	✓ ✓	C_2H_5OH	✓	✓
CH_3CN-2	✓	✓	✓	✓	✓	✓	✓	✓	✓	C_2H_5OH	C_2H_5OH	✓	✓	C_2H_5OH	✓	✓
CH_3CN-4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	C_2H_5OH	✓	✓
CH_3CN-8	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	C_2H_5OH	✓	C_2H_5OH	✓	✓
CH_3CHO-0.5	C_2H_5OH	CH_3OH	C_2H_5OH	C_2H_5OH	C_2H_5OH	CH_3OH	C_2H_5OH	C_2H_5OH	C_2H_5OH	✓	✓	✓	C_2H_5OH	HCOOH	C_2H_5OH	C_2H_5OH
CH_3CHO-1	✓	✓	✓	✓	✓	✓	✓	CH_3OH	✓	✓	✓	✓	C_2H_5OH	✓	✓	✓
CH_3CHO-2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CH_3CHO-1	CH_3OH	CH_3OH	HNO_3	C_2H_5OH	C_2H_5OH	✓	CH_3OH	C_2H_5OH	CH_3OH	H_2S	✓	✓	✓	✓	✓	✓

Table 4 (continued)

Classifier type	Regular classifiers								One-vs-Rest (OVR) classifiers							
Experimental spectrum	k-NN	DT	RF	SVM -linear	SVM -RBF	MLP	ABDT	ABRF	k-NN	DT	RF	SVM -linear	SVM -RBF	MLP	ABDT	ABRF
CH_3CHO-2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CH_3CHO-8	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Accuracy	80.56	38.89	52.78	63.89	75.00	94.44	77.78	75.00	77.78	36.11	55.56	94.44	88.89	75.00	83.33	83.33

One random trial is shown here. See supplementary material for the other two cases. A check mark (✓) indicates correct classification and a chemical compound indicates incorrect classification, where the listed compound is the classifier's incorrect prediction. A hyphenated number appearing after the chemical formula indicates pressure in Torr

Table 5 Overall average and standard deviation values for classification accuracy, precision, recall, and F1 score for 36 experimental spectra

Methods	Accuracy [%]	Precision	Recall	F1 score
k-NN	80.56 ± 2.78	0.91 ± 0.14	0.81 ± 0.14	0.84 ± 0.08
DT	50.93 ± 11.23	0.60 ± 0.14	0.51 ± 0.23	0.49 ± 0.19
RF	51.85 ± 1.61	0.67 ± 0.37	0.52 ± 0.33	0.55 ± 0.30
SVM-linear	66.67 ± 4.81	0.74 ± 0.42	0.67 ± 0.35	0.67 ± 0.34
SVM-RBF	76.85 ± 3.21	0.92 ± 0.17	0.77 ± 0.18	0.80 ± 0.12
MLP	87.04 ± 6.99	0.86 ± 0.18	0.87 ± 0.15	0.80 ± 0.24
ABDT	75.00 ± 4.82	0.85 ± 0.11	0.75 ± 0.16	0.78 ± 0.10
ABRF	73.15 ± 8.49	0.89 ± 0.18	0.73 ± 0.24	0.76 ± 0.14
OVR(k-NN)	78.71 ± 1.61	0.91 ± 0.14	0.79 ± 0.13	0.83 ± 0.07
OVR(DT)	42.59 ± 5.78	0.57 ± 0.40	0.43 ± 0.35	0.39 ± 0.24
OVR(RF)	57.41 ± 5.78	0.76 ± 0.18	0.58 ± 0.33	0.59 ± 0.25
OVR(SVM-linear)	93.52 ± 1.60	0.96 ± 0.10	0.94 ± 0.13	0.94 ± 0.09
OVR(SVM-RBF)	87.04 ± 3.21	0.94 ± 0.17	0.87 ± 0.13	0.89 ± 0.09
OVR(MLP)	75.93 ± 4.25	0.70 ± 0.41	0.76 ± 0.38	0.72 ± 0.38
OVR(ABDT)	75.92 ± 6.41	0.92 ± 0.07	0.76 ± 0.23	0.80 ± 0.14
OVR(ABRF)	75.92 ± 6.41	0.92 ± 0.07	0.76 ± 0.23	0.80 ± 0.14

Values are averaged over 3 random trials

different spectra for two compounds (ethanol and acetonitrile). Interestingly, OVR(SVM-RBF) misclassified four spectra for five different compounds (nitric acid, ethanol, chloromethane, acetonitrile and acetaldehyde).

3.3 Misclassifications

Misclassifications of experimental spectra appear to occur for two reasons. The first being deficiencies in the simulated training data set used to train the classifiers. The ML classifiers are highly sensitive to the frequency location of spectral features (absorption peaks) and the relative strength of spectral features. If experimental measurements are slightly shifted in frequency space, relative to simulated training spectra, or contain features that can be confused with non-corresponding simulated training spectra due to slight mismatches in frequencies of peaks or the relative absorption of peaks, they are more likely to be misclassified. This sensitivity, in part, motivates the lower frequency resolution used to develop the spectral simulation database, as described above. However, even with the lower frequency resolution of the simulated spectral training data set, misclassifications can occur due to differences between simulated training spectra and experimental measurements. It is important to note, that the simulated training spectra are generated from parameters in spectroscopy databases that are not fully validated by experiments, especially in the present frequency range. In several cases, the experimental data used in this study is the first of its kind. The second reason for misclassifications is experimental noise. Depending on the relative strength of measured spectral features to the underling absorption noise, classifiers will sometimes attempt to "fit" the noise and "see" noise features as spectral features, causing a misclassification, a phenomena known as "overfitting" in ML terminology.

Figure 11 illustrates an experimental spectrum for formic acid at 1 Torr that is misclassified by the MLP classifier as methanol. The classifiers are designed to only predict labels (compounds) and not pressure but for comparison we have plotted the simulated spectra for both formic acid and methanol at 1 Torr against the experiment. The experimental measurement for this weak absorption case shows four main absorption features with relatively poor signal-to-noise. A comparison of the formic acid simulation with the experimental spectra shows that there are slight differences between the two. The relative magnitude of the four strongest absorption features in the simulation are not in agreement with the experiment and there are some slight differences in the frequencies of the peaks. The simulated spectra for methanol have 5–6 spectral features that match local absorbance peaks in the experimental spectra, although not all of the strongest four features. It is likely that because the methanol training simulation matches a greater number of local peaks, even though they are generally weaker spectral features in the experiment, the classifier incorrectly finds methanol to be a better match. The poor signal-to-noise magnifies this problem causing the classifier to have difficulty identifying the important fingerprint features of the experimental spectrum. From the tenfold cross-validation results shown in Fig. 10, we also observe that classification accuracy is lower at pressures below 1 Torr where the signal-to-noise is reduced due to weak absorption signals.

4 Conclusions

The present work demonstrates the classification of pure component absorption spectra in the THz spectral region using machine learning (ML) methods and evaluates the relative performance of a variety of ML classifiers for the classification of simulated and experimental spectra. ML classifiers were trained using simulated spectra for twelve pure compounds in the 220–330 GHz range generated using fundamental spectroscopic parameters. The spectra present

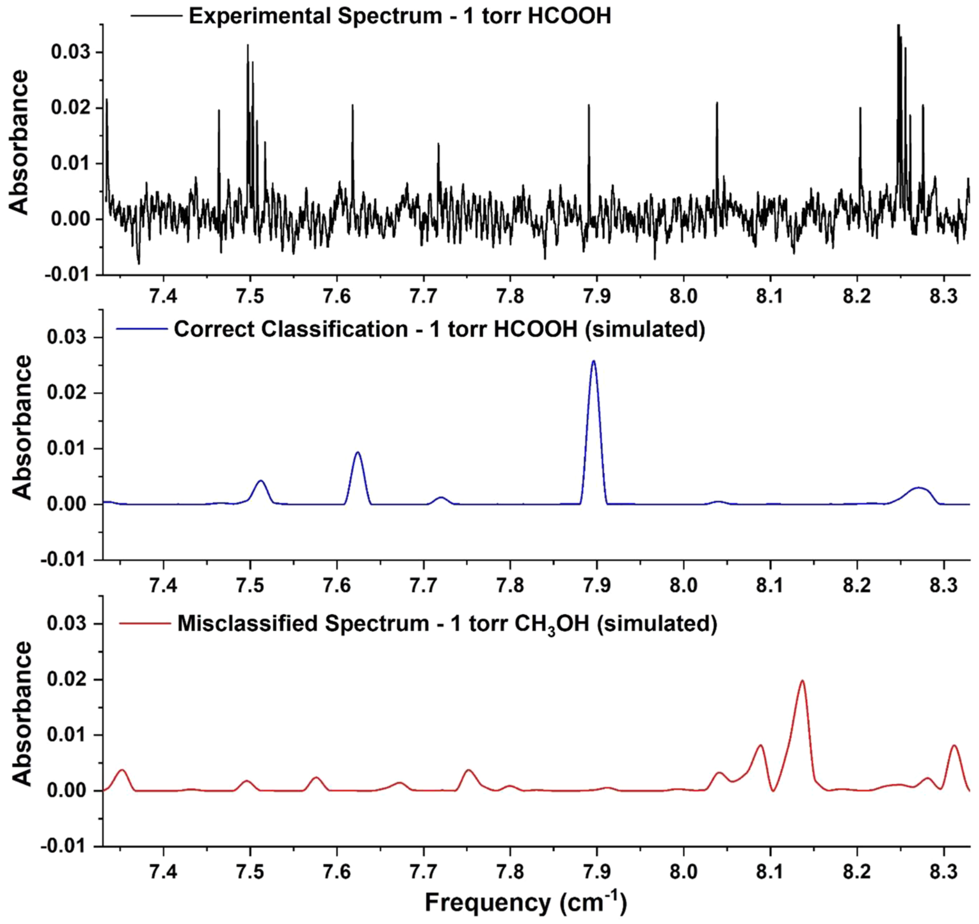


Fig. 11 Spectral demonstration of the misclassification of formic acid at 1 Torr by MLP classifier

complex fingerprints based on the rotational energy level structure for each polar compound considered. Classifier performance was first evaluated against simulated spectra, in both a 70–30% training–testing split and in tenfold cross-validation studies, and then against 36 measured spectra for six compounds in the same 220–330 GHz range. The ML classifiers considered include *k*-nearest neighbors, decision trees, random forest, support vector machines (with linear and radial basis function kernels), multi-layer perceptron, adaptive boosted decision trees and random forest, and one-vs-rest implementation of the aforementioned ML methods. Compounds considered include polar compounds of industrial and environmental importance for which gas sensing may be desired: chloromethane, methanol, formic acid, formaldehyde, hydrogen sulfide, sulfur dioxide, carbonyl sulfide, hydrogen cyanide, acetonitrile, nitric acid, ethanol, and acetaldehyde.

All classifiers perform extremely well in identifying simulated spectra (accuracy > 99%); however, when presented with experimental data containing noise, the multi-layer perceptron (MLP) and the one-vs-rest implementation of the support vector machine with both linear kernel and radial basis kernel function (OVR(SVM-linear) and OVR(SVM-RBF)) classifiers achieved an average classification accuracy of greater than 85% on a set of experimental absorption spectra for six compounds and high recall (87% for MLP and OVR(SVM-RBF) and 94% for OVR(SVM-linear). Misclassifications generally occur for situations involving weak spectral features, where noise compromises the classification.

The novelty of the present work is the demonstration of automated ML-based spectral fingerprinting within a narrow frequency range in the THz region for noisy experimental data and with relatively high accuracy and fast training times, that are suitable for real-time gas sensing and continuous training of classifiers. The methods demonstrated here can be directly extended to different spectral frequency domains, larger and/or different databases of compounds, and a wide variety of conditions (pressure and temperatures). Extending the present ML classification methods for the identification of multi-component mixture spectra will require future work but is possible.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00340-021-07582-0.

Acknowledgements This work was supported by the National Science Foundation under Grant CBET-1851291.

References

1. R.H. Jacobsen, D.M. Mittleman, M.C. Nuss, Opt. Lett. **21**, 2011 (1996)
2. C.N. Banwell, E.M. McCash, *Fundamentals of Molecular Spectroscopy*, 4th edn. (McGraw-Hill Education, New York, 2016).
3. G. Herzberg, *Molecular Spectra and Molecular Structure II. Infrared and Raman Spectra of Polyatomic Molecules*, 1st edn. (D. Van Nostrand Company Inc., New York, 1945).
4. H.W. Kroto, *Molecular Rotation Spectra*, 1st edn. (Wiley, Hoboken, 1975).
5. C.H. Townes, A.L. Schawlow, *Microwave Spectroscopy*, 1st edn. (McGraw-Hill Book Company Inc., New York, 1955).
6. P. Bunker, P. Jensen, *Molecular Symmetry and Spectroscopy*, 2nd edn. (NRC Research Press, Ottawa, 1998).
7. T.E. Rice, M.A.Z. Chowdhury, M.W. Mansha, M.M. Hella, I. Wilke, M.A. Oehlschlaeger, Appl. Phys. B: Lasers Optics **126**, 152 (2020)
8. M. W. Mansha, K. Wu, T. E. Rice, M. A. Oehlschlaeger, M. M. Hella, and I. Wilke, Proceedings of IEEE Sensors 3 (2019).
9. A. Tekawade, T. E. Rice, M. A. Oehlschlaeger, M. W. Mansha, K. Wu, M. M. Hella, and I. Wilke, in *International Conference on Infrared, Millimeter, and Terahertz Waves, IRMMW-THz* (2019).
10. M. Naftaly, N. Vieweg, A. Deninger, Sensors **19**, 4203 (2019)
11. F. Elmaz, B. Büyükçakır, Ö. Yücel, A.Y. Mutlu, Fuel **266**, 117066 (2020)
12. X. Cui, Q. Wang, Y. Zhao, X. Qiao, G. Teng, Appl. Phys. B: Lasers Optics **125**, 1 (2019)
13. H. Hao, R. Guo, Q. Gu, X. Hu, Miner. Eng. **143**, 105899 (2019)
14. O. Gazeli, E. Bellou, D. Stefas, S. Couris, Food Chem. **302**, 125329 (2020)
15. E. Bellou, N. Gyftokostas, D. Stefas, O. Gazeli, S. Couris, Spectrochimica Acta - Part B Atomic Spectroscopy **163**, 105746 (2020)
16. R.M. Balabin, R.Z. Safieva, Anal. Chim. Acta **689**, 190 (2011)
17. O. Egorova, R. Hafizi, D.C. Woods, G.M. Day, J. Phys. Chem. A **124**, 8065 (2020)
18. B.X. Xue, M. Barbatti, P.O. Dral, J. Phys. Chem. A **124**, 7199 (2020)
19. M.A. Cusentino, M.A. Wood, A.P. Thompson, J. Phys. Chem. A **124**, 5456 (2020)
20. Y. Zuo, C. Chen, X. Li, Z. Deng, Y. Chen, J. Behler, G. Csányi, A.V. Shapeev, A.P. Thompson, M.A. Wood, S.P. Ong, J. Phys. Chem. A **124**, 731 (2020)
21. M.G. Taylor, T. Yang, S. Lin, A. Nandy, J.P. Janet, C. Duan, H.J. Kulik, J. Phys. Chem. A **124**, 3286 (2020)
22. P. Rowe, G. Csányi, D. Alfè, A. Michaelides, Phys. Rev. B **97**, 054303 (2018)
23. T. Kavzoglu, I. Colkesen, Int. J. Appl. Earth Obs. Geoinf. **11**, 352 (2009)
24. E. Antono, N.N. Matsuzawa, J. Ling, J.E. Saal, H. Arai, M. Sasago, E. Fujii, J. Phys. Chem. A **124**, 8330 (2020)
25. P. Peng, X. Zhao, X. Pan, W. Ye, Sensors (Switzerland) **18**, 1 (2018)
26. X. Zhai, A.A.S. Ali, A. Amira, F. Bensaali, IEEE Access **4**, 8138 (2016)
27. F. Benrekia, M. Attari, M. Bouhedda, Sensors (Switzerland) **13**, 2967 (2013)
28. C. Cortes and V. Vapnik, Patent no. US5640492A (1997).
29. A.E. Maxwell, T.A. Warner, F. Fang, Int. J. Remote Sens. **39**, 2784 (2018)
30. M. Pardo, G. Sberveglieri, Sens. Actuat. B: Chem. **107**, 730 (2005)
31. Ł Lentka, J.M. Smulko, R. Ionescu, C.G. Granqvist, L.B. Kish, Metrol. Measure. Syst. **22**, 341 (2015)
32. S. Güney, A. Atasoy, Sens. Actuat. B: Chem. **166–167**, 721 (2012)
33. J.H. Cho, P.U. Kurup, Sens. Actuat. B: Chem. **160**, 542 (2011)
34. H. Tian, H. Liu, Y. He, B. Chen, L. Xiao, Y. Fei, G. Wang, H. Yu, C. Chen, J. Food Measure. Characteriz. **14**, 573 (2020)

35. Y. Luo, W. Ye, X. Zhao, X. Pan, Y. Cao, Sensors (Switzerland) **17**, 1 (2017)
36. F. Masulli, M. Pardo, G. Sberveglieri, and G. Valentini, in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2002).
37. J. Mingers, Mach. Learn. **4**, 227 (1989)
38. C.F. Neese, I.R. Medvedev, G.M. Plummer, A.J. Frank, C.D. Ball, F.C. De Lucia, IEEE Sens. J. **12**, 2565 (2012)
39. B.M. Fischer, H. Helm, P.U. Jepsen, Proc. IEEE **95**, 1592 (2007)
40. R.M. Smith, M.A. Arnold, Anal. Chem. **87**, 10679 (2015)
41. A. Tekawade, T.E. Rice, M.A. Oehlschlaeger, M.W. Mansha, K. Wu, M.M. Hella, I. Wilke, Appl. Phys. B: Lasers Optics **124**, 105 (2018)
42. I.E. Gordon, L.S. Rothman, C. Hill, R.V. Kochanov, Y. Tan, P.F. Bernath, M. Birk, V. Boudon, A. Campargue, K.V. Chance, B.J. Drouin, J.M. Flaud, R.R. Gamache, J.T. Hodges, D. Jacquemart, V.I. Perevalov, A. Perrin, K.P. Shine, M.A.H. Smith, J. Tennyson, G.C. Toon, H. Tran, V.G. Tyuterev, A. Barbe, A.G. Császár, V.M. Devi, T. Furtenbacher, J.J. Harrison, J.M. Hartmann, A. Jolly, T.J. Johnson, T. Karman, I. Kleiner, A.A. Kyuberis, J. Loos, O.M. Lyulin, S.T. Massie, S.N. Mikhailenko, N. Moazzen-Ahmadi, H.S.P. Müller, O.V. Naumenko, A.V. Nikitin, O.L. Polyansky, M. Rey, M. Rotger, S.W. Sharpe, K. Sung, E. Starikova, S.A. Tashkun, J. Vander Auwera, G. Wagner, J. Wilzewski, P. Wcisło, S. Yu, E.J. Zak, J. Quant. Spectrosc. Radiat. Transfer **203**, 3 (2017)
43. H.M. Pickett, R.L. Poynter, E.A. Cohen, M.L. Delitsky, J.C. Pearson, H.S.P. Müller, J. Quant. Spectrosc. Radiat. Transfer **60**, 883 (1998)
44. R.V. Kochanov, I.E. Gordon, L.S. Rothman, P. Wcisło, C. Hill, J.S. Wilzewski, J. Quant. Spectrosc. Radiat. Transfer **177**, 15 (2016)
45. G. Van Rossum, *Python Reference Manual* (Amsterdam, 1995).
46. G. Hinton and S. Roweis, in *Advances in Neural Information Processing Systems* (2003).
47. L. van der Maaten, G. Hinton, J. Mach. Learn. Res. **1**, 1 (2008)
48. Y. S. Abu-Mostafa, M. Magdon-Ismail, and H. T. Lin, *Learning from data: a short course* (AMLBook, 2012).
49. C.M. Bishop, *Machine Learning and Pattern Recoginiton* (Springer, New York, 2006).
50. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay, J. Mach. Learn. Res. **12**, 2825 (2011)
51. B.W. Silverman, M.C. Jones, Int. Stat. Rev./Revue Internationale de Statistique **57**, 233 (1989)
52. T.M. Cover, P.E. Hart, IEEE Trans. Inf. Theory **13**, 21 (1967)
53. M.E. Hellman, IEEE Trans. Syst. Sci. Cybernet. **6**, 179 (1970)
54. K. Fukunaga, L.D. Hostetler, IEEE Trans. Inf. Theory **21**, 285 (1975)
55. T. Bailey, A.K. Jain, IEEE Trans. Syst. Man Cybernet. **SMC-8**, 311 (1978)
56. J.E.S. Macleod, A. Luk, D.M. Titterington, IEEE Trans. Syst. Man Cybernet. **17**, 689 (1987)
57. L. Peterson, DOI: https://doi.org/10.4249/Scholarpedia.1883 (2009).
58. K. Chomboon, P. Chujai, P. Teerarassammee, K. Kerdprasop, and N. Kerdprasop, in *International Conference on Industrial Application Engineering* (2015).
59. O. Kramer, in *Proceedings - 10th International Conference on Machine Learning and Applications, ICMLA 2011* (2011).
60. S. Salzberg, Mach. Learn. **16**, 235 (1993)
61. L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, *Classification and Regression Trees* (Chapman & Hall/CRC, Boca Raton, 1984).
62. J.R. Quinlan, Machine Learning **1**, 81 (1986)
63. J.R. Quinlan, *C4.5: Programs for Machine Learning* (Springer, San Mateo, 1993).
64. Scikit-learn 0.23.2 documentation, Scikit-Learn (2020).
65. L. Breiman, Mach. Learn. **45**, 5 (2001)
66. P. Geurts, D. Ernst, L. Wehenkel, Mach. Learn. **63**, 3 (2006)
67. C. Cortes, V. Vapnik, Mach. Learn. **20**, 273 (1995)
68. J. Weston and C. Watkins, Citeseer: Technical Report 23 (1998).
69. A.J. Smola, B. Scholkopf, Stat. Comput. **14**, 199 (2004)
70. G. Anthony, H. Gregg, and M. Tshilidzi, in *28th Asian Conference on Remote Sensing 2007, ACRS 2007* (2007).
71. J. Shawe-Taylor and S. Sun, Academic Press Library in Signal Processing: Volume 1 Signal Processing Theory and Machine Learning **1**, 857 (2014).
72. C. Hsu, C. Chang, and C. Lin, National Taiwan University 1396 (2003).
73. S.S. Keerthi, C.J. Lin, Neural Comput. **15**, 1667 (2003)
74. S. Haykin, Soft Comput. Intell. Syst. 71 (2000).
75. H.S. Hippert, C.E. Pedreira, R.C. Souza, IEEE Trans. Power Syst. **16**, 44 (2001)
76. J. Leonard, M.A. Kramer, Comput. Chem. Eng. **14**, 337 (1990)
77. P. J. Werbos, PhD Thesis, Harvard University (1974).
78. Y. Freund, R.E. Schapire, J. Comput. Syst. Sci. **55**, 119 (1997)
79. Y. Freund, Inf. Comput. **121**, 256 (1995)
80. Y. Freund, R. Schapire, J. Jpn. Soc. Artif. Intell. **14**, 771 (1999)
81. G. Anthony, H. Gregg, and M. Tshilidzi, 28th Asian Conference on Remote Sensing 2007, ACRS 2007 **2**, 801 (2007).
82. J.D. Rodríguez, A. Pérez, J.A. Lozano, IEEE Trans. Pattern Anal. Mach. Intell. **32**, 569 (2010)
83. T. Dietterich, ACM Comput. Surv. (CSUR) **27**, 326 (1995)