Journal of Mass Spectrometry and Advances in the Clinical Lab 23 (2022) 1–6

Contents lists available at ScienceDirect

Journal of Mass Spectrometry and Advances in the Clinical Lab

journal homepage: www.sciencedirect.com/journal/journal-of-mass-spectrometry-and-advances-in-the-clinical-lab

Supervised machine learning in the mass spectrometry laboratory: A tutorial

Edward S. Lee [a,b], Thomas J.S. Durant [a,b,*]

[a] Department of Laboratory Medicine, at Yale School of Medicine, New Haven, CT, USA
[b] Department of Laboratory Medicine, at Yale New Haven Hospital, New Haven, CT, USA

ARTICLE INFO

Keywords:
Supervised machine learning
Xgboost
Artificial intelligence
Mass spectrometry
Amino acid

ABSTRACT

As the demand for laboratory testing by mass spectrometry increases, so does the need for automated methods for data analysis. Clinical mass spectrometry (MS) data is particularly well-suited for machine learning (ML) methods, which deal nicely with structured and discrete data elements. The alignment of these two fields offers a promising synergy that can be used to optimize workflows, improve result quality, and enhance our understanding of high-dimensional datasets and their inherent relationship with disease. In recent years, there has been an increasing number of publications that examine the capabilities of ML-based software in the context of chromatography and MS. However, given the historically distant nature between the fields of clinical chemistry and computer science, there is an opportunity to improve technological literacy of ML-based software within the clinical laboratory scientist community. To this end, we present a basic overview of ML and a tutorial of an ML-based experiment using a previously published MS dataset. The purpose of this paper is to describe the fundamental principles of supervised ML, outline the steps that are classically involved in an ML-based experiment, and discuss the purpose of good ML practice in the context of a binary MS classification problem.

Introduction

As the demand for laboratory testing by mass spectrometry (MS) increases, so does the need for automated methods for data analysis. The alignment of machine learning (ML) and MS offers a promising synergy that can be leveraged to optimize workflows, improve quality assurance practices, and enhance our clinical understanding of high-dimensional (i.e., multivariate) datasets. To this end, there has been an increasing number of publications that explore the capabilities of ML-based software that optimize clinical chromatography and MS workflows. However, given the historically distant nature between the fields of clinical chemistry and artificial intelligence, there is an opportunity to improve technological literacy among the clinical laboratory scientist community around ML-based software [1]. Knowledge gaps related to the fundamental principles of ML can hinder the intuitive understanding of how these novel applications function, and as a result, limit our ability to ensure ML-based technology is appropriately validated and subjected to ongoing quality assurance procedures. In this report, we use a previously published dataset of categorically labeled plasma amino acid (PAA) profiles to provide a high-level tutorial of an ML-based experiment, to be used as a contextual framework for exploring the fundamental principles and workflow of ML model development.

Supervised machine learning

There are three, commonly referenced categories of ML, which are supervised learning, unsupervised learning, and reinforcement learning (RL) [2,3]. In the context of ML, supervision refers to 'labels' that are individually paired with input data (e.g., PAA profiles labeled as 'normal' or 'abnormal'). In classification problems, these labels are often assigned by human experts, and the labor required to make these labels is a key barrier to ML development. The goal of supervised ML is to model a relationship between input data (e.g., a patient's PAA profile) and output data (e.g., is the PAA profile normal or abnormal) using a dataset of input–output pairs. Unsupervised ML involves datasets with no human annotation, and the identification of patterns is unbound by

Abbreviations: CART, Classification and Regression Trees; NLL, Negative Log Loss; ML, Machine Learning; MS, Mass Spectrometry; PAA, Plasma Amino Acid; PR, Precision-Recall; PRAUC, Area Under the Precision-Recall Curve; RL, Reinforcement Learning; ROC, Receiver Operator Curve; SCF, Supplemental Code File; XGBT, Extreme Gradient Boosted Trees.

* Corresponding author at: Department of Laboratory Medicine, 55 Park Street PS345D, New Haven, CT 06511, USA.
E-mail address: thomas.durant@yale.edu (T.J.S. Durant).

https://doi.org/10.1016/j.jmsacl.2021.12.001
Received 14 July 2021; Received in revised form 2 December 2021; Accepted 6 December 2021
Available online 13 December 2021

the existence of labels [3]. These methods are often used as approaches to data-driven knowledge discovery. RL is a less commonly applied method that implies the use of reward and penalty signals to drive learning. The purpose of this article is to provide an introductory overview of supervised ML, as this is the most encountered form of ML in the clinical laboratory today. For the interested reader, more comprehensive overviews of artificial intelligence, unsupervised ML, and RL can be found in recent publications [3,4].

In supervised ML, datasets are comprised of: [1] data (input, commonly denoted as *X*) and [2] labels (output, commonly denoted as *Y*). Labels are typically categorical or continuous variables. In this paper, we use a previously published dataset of PAA profiles with individual, categorical labels of 'normal' or 'abnormal'. The following sections follow a traditional approach to an ML-based binary classification problem that begins with identifying the clinical need. Subsequent sections will progress through the customary heuristics of ML development, including preparation of the dataset, choosing the appropriate ML method, training of the algorithm, and testing the model [4].

Binary classification with machine learning

Hardware and software

Computation for model training and testing was performed on a commodity laptop running a macOS (version: 10.15.7). Processing hardware included 1 CPU (Intel(R) Core i7 I7-9750H @ 2.6 GHz). Machine learning functions that generated the results presented in the primary manuscript were implemented in parallel using both R (version: 4.1.0) and Python (version: 3.9.4) [5–8]. Results generated in R and Python are comparably similar but not identical. For this reason, only the results from the R implementation are presented in the following sections. To help illustrate the implementation of ML, R and Python code that is relevant to concepts presented in the following sections can be viewed in the supplemental material. These documents are referenced as supplementary code files (SCF). For readers interested in running the analysis and interacting with the data, the complete code used to perform these analyses is also available for public download here [9,10]. The dataset is available for download in the supplementary material of Wilkes et al. [11].

Identifying the clinical need

Data generated by clinical MS assays are particularly well-suited for ML-based methods which deal nicely with structured, high-dimensional datasets [2]. Accordingly, there are many opportunities to implement ML-based solutions in the clinical MS laboratory. This is reflected in recent publications that examine the use of ML for peak integration [12,13], autoverification [14,15], clinical interpretation support (CIS) [11,16,17], and the support of novel MS applications at the point of care [18,19]. For the purposes of this paper, and the sake of simplicity, we describe the development of a model that can predict whether a PAA profile is 'normal' or 'abnormal'. With only two possible outputs (classes), this is known as a binary classification ML task.

Preparing the dataset

Datasets can be classified as structured, semi-structured, or unstructured. A dataset is structured if there is an inherent organizational paradigm (i.e., model), that provides a framework to access individual data elements. Unstructured datasets contain no inherent organization of data elements; examples include digital images or raw DNA sequencing data. The dataset used in Wilkes et al. is structured, as it contains discrete data elements that are composed in rows and columns and can be stored as a delimited file with a comma separator (SCF Code Block 2).

Prior to training, data preprocessing is often required to make ML analysis feasible. The Wilkes dataset comprises 2,084 PAA profiles, 31% (n = 644) of which are labeled as abnormal. Each PAA profile consists of 24 continuous variables and 4 categorical variables. To this end, categorical variables (e.g., Sex) were encoded as numeric values since a numeric representation of the data is required by the ML algorithm that will be used (Fig. 1) (SCF Code Block 3). It should be noted that Wilkes et al. already performed preprocessing on the data and made specific decisions for handling censored data, such as setting undetectable levels to 0 and replacing values of "Present," "Detected," or "Trace," with a fixed, arbitrarily low value. Commenting on these choices is outside the scope of this paper, but these kinds of decisions can affect the performance of ML algorithms and must be made carefully. Other forms of data preprocessing, such as imputation of missing values or normalization of numeric variables, were not performed in this tutorial. In datasets with missing values, missing value imputation can be performed by using statistical techniques to make inferences on values to substitute missing data in order to avoid throwing out data, which can cause biases in the ML model [20,21]. Normalization should be performed when values of numeric variables have different orders of magnitude since ML algorithms usually need inputs on similar scales [22]. These forms of data preprocessing can be implemented for tuning algorithm performance and are outside the scope of this paper.

The purpose of supervised ML in binary classification is to train a model that can make accurate class predictions (i.e., normal or abnormal) on novel input data – i.e., data that was not analyzed during training. Accordingly, the total dataset (n = 2084) is divided into separate groups, commonly using a random 70:30 split, creating independent train and test datasets, respectively (SCF Code Block 4). The test dataset is purposefully set aside and not presented to the algorithm while training the model. This is referred to as the 'holdout method'. The train dataset is then split into a training set and a validation set. The resulting smaller training set is the sample of data used to fit the model, and the validation set is used to evaluate the model fit after each iteration of training. It is important to note that splitting data sets should be done randomly in order to avoid biasing the ML model with unforeseen effects that can occur with manual splitting, and accordingly, helper functions in machine learning packages exist to facilitate appropriate splitting (SCF Code Block 4).

Conceptually, the train-test split is analogous to providing a group of students with practice questions (i.e., the training dataset), but not allowing them to see the final exam questions (i.e., the testing dataset). This practice avoids an overly optimistic assessment of how well the model learned during training and provides a more conservative and accurate estimate of performance on future, unseen data.

Dividing the dataset for training and testing can be approached in several ways but is commonly done as a one-time split or many splits in what is known as *k*-fold cross-validation (CV). The latter involves repeating the training and testing process using *k* number of random splits and taking the mean of the observed performance metrics. As opposed to a one-time split, *k*-fold CV provides *k* independent calculations of test metrics that can be used to derive a range of expected performance that accounts for variability that is derived from random division of train and test datasets. However, in this paper, for the sake of simplicity and ease of interpreting the supplemental material, we performed a one-time 70:30 split of the original dataset. This was done using a stratified shuffle split function that maintains an approximately equal distribution of 'positives' between the train and test datasets. Accordingly, post-split, there were 30% (n = 352) and 33% (n = 204) abnormal PAA profiles in the train and test datasets, respectively (Fig. 1) (SCF Code Block 4).

Choosing the algorithm

Once the clinical need is defined and a dataset is appropriately curated, an ML algorithm can be selected. An algorithm is the mathe-

ID	Sex	Tau	Thr	Ser	…	Leu.Ala	Class
1	M	174	55	85	…	0.102	Abnormal
2	M	94	111	126	…	0.198	Normal
3	F	56	54	107	…	0.127	Abnormal
4	F	167	90	141	…	0.407	Normal
5	F	50	50	65	…	0.111	Abnormal
…	…	…	…	…	…	…	…
2083	M	71	143	161	…	0.214	Normal
2084	F	96	214	182	…	0.345	Normal

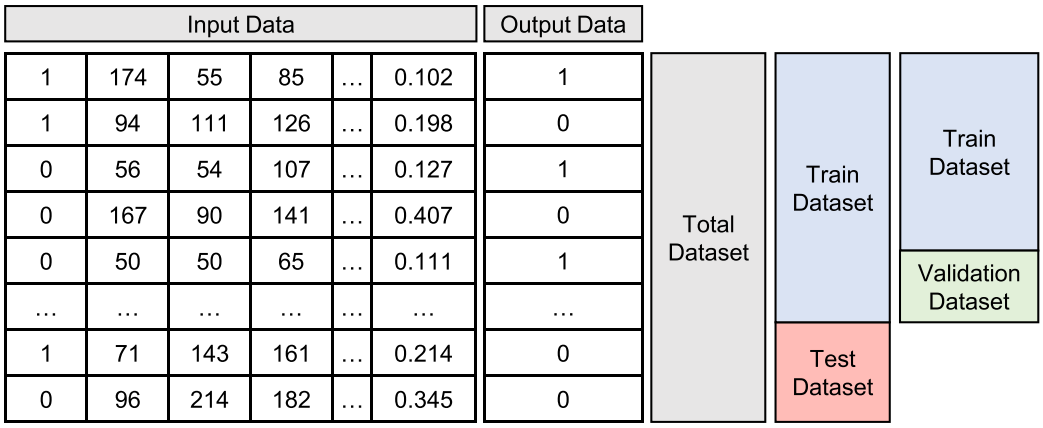

Fig. 1. Data preprocessing schematic. (Top) Original PAA profile dataset represented in a tabular format. This is an example of structured dataset. (Bottom) Post-preprocessing: Identifier column is removed, and categorical data is encoded as numeric values. In addition, the data used for making predictions (input) is separated from the labels (output). This is done so that the labels are not used by the algorithm as a feature for making predictions during training. (Bottom right) Lastly, input and output data are separated into train, validation, and test datasets.

matical machinery used to make a prediction (denoted as $\hat{A}\cdot$) from the input data (denoted as x_i). Logistic regression is a well-known example of an algorithm that can be used for binary classification and, while it is often grouped with traditional statistical methods, it is commonly used in ML. There are many ML algorithms in existence and choosing the right algorithm to support an application is not always straightforward. Currently, best practice in ML would recommend choosing the 'correct' algorithm based on the intended task and on technical considerations, such as the input and output data type and content [23]. In addition, like choosing the parameters for chromatography and MS development, ML method selection protocols often rely on trial and error with several types of algorithms.

For the purposes of this experiment, we chose to use an implementation of extreme gradient boosted trees (XGBT) (SCF Code Block 5) [24,25]. Given the multitude of factors that can influence algorithm selection, a comprehensive evaluation and reasoning for using XGBT cannot be fully addressed within the intended scope of this paper. Briefly, XGBT is based on a combination of gradient boosting and decision trees; for the latter, although there are many approaches, classification and regression trees (CART) are most often used [24,26]. While there are few studies that empirically evaluate modern ML classifiers, available literature suggest that boosted trees are among the best, 'off the shelf' classifiers to date, based on various performance metrics [23,27]. Accordingly, XGBT is widely used in ML research today. It was also one of the algorithms used by Wilkes et al. in the work originally associated with the PAA profile dataset used in this paper.

Training the algorithm

ML training is the process of iterative calculations over a group of algebraic functions in a specific sequence. Conceptually, each iteration involves the following steps: present a batch of training data to the algorithm, make predictions on all samples in the batch, calculate a loss function, and update model parameters based on the loss value (Fig. 2). The parameters are the variables in the algorithm that, when adjusted, affect the accuracy of predictions. Updates to parameters are guided by calculating loss (i.e., error) using a loss function. A loss function is designed to measure the mathematical distance between the true and predicted labels. Customarily, as in this experiment, the loss is proportional to the amount of disagreement between the predicted label and the true label; therefore, the smaller the loss, the more accurate the prediction. Accordingly, machines learn by comparing predicted labels with true labels and adjust the model parameters to minimize that margin of error. The appropriate choice for a loss function, and for the mathematical details of how to calculate the "disagreement" between the prediction and the labels, depends on the type of problem being answered and the ML algorithm being used. For example, a binary classification problem could use the negative log likelihood of a statistical distribution, and a regression problem—where the model is making a numeric value prediction based on input data—usually uses the root mean square error. The process of selecting loss functions must consider the questions being asked and the type of learning being performed on the data.

Loss calculations are performed on the training dataset and a second 'hold out dataset'. In addition to the train-test split, the training dataset

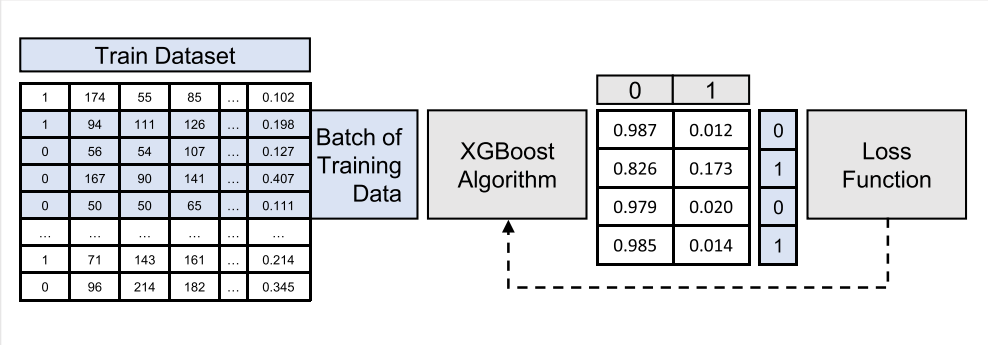

Fig. 2. Machine learning workflow schematic for binary classification of PAA profiles as 'normal' or 'abnormal'. (From left to right) A sample (batch) of training data is presented to the XGBoost algorithm. XGBoost analyses the data and makes predictions for each class, 'normal' and 'abnormal', and represents the predictions as a sum-to-one probability distribution, with probability of 'normal' in the left column and 'abnormal' in the right column of the prediction table. The true labels are then compared against the predicted classes via a loss function. The calculated loss is then used to update the parameters of the XGBoost algorithm. This process repeats itself until the loss no longer decreases.

undergoes an additional, and final, 80:20 split to generate a training dataset and a validation dataset (Fig. 1) (SCF Code Block 4). After each iteration, predictions are made on a batch of training and validation samples and checked against the true labels. From those comparisons, the mean loss is calculated across all training and validation samples and referred to as the training loss and validation loss. The training loss is used to guide updates to algorithm parameters. The validation loss is used to monitor for overfitting. If the ML algorithm is learning appropriately, the training and validation loss should decrease across training iterations. Once the validation loss is no longer decreasing over a predefined number of training iterations, the algorithm predictions are unlikely to improve further. At this point, training should be stopped to avoid the risk of overfitting the model. For a more detailed description of overfitting, please refer to the review by Harrison et al [3].

In this experiment, the final training protocol was as follows (SCF Code Block 6). Following the 80:20 split, the training and validation datasets were comprised of 1,166 and 292 samples, with 30% (n = 352) and 33% (n = 88) abnormal cases, respectively. Negative log-likelihood (NLL) was used as the validation loss function. After the first training iteration, validation loss was 0.639. At iteration 82, validation loss had not improved in the last ten iterations and training stopped. Accordingly, model parameters from iteration 72 were saved, where a minimum validation loss of 0.214 was achieved. Lastly, the training and validation loss curves were visually assessed for overfitting, which would be indicated by positive divergence of the validation loss from the training loss. This divergence suggests overfitting since the model is fitting to the training set but failing to generalize to the validation set, and thus choosing model parameters from the iteration where this divergence begins would avoid overfitting. Visual inspection of training and validation loss curves indicate minimally appreciable divergence up to iteration 72, at which point model parameters were saved (Fig. 3) (SCF Code Block 6).

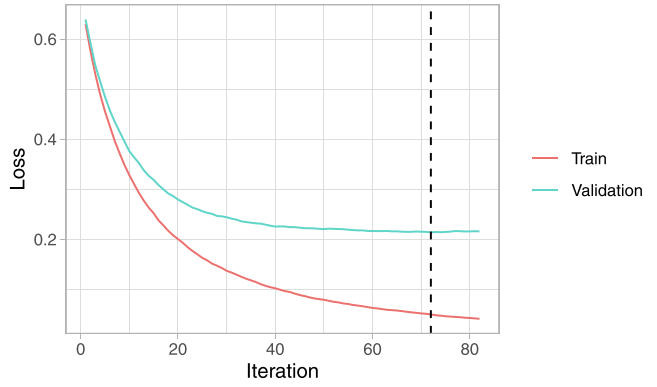

Fig. 3. Training and validation loss plotted as a function of training iterations.

Testing the model

During training, the parameters in the algorithm are updated to achieve the most accurate predictions. Once those parameters are identified, they are saved into a file that is referred to as the model. Once a model is saved, it will not change unless additional training is performed, and new parameters are saved over it. Models are saved in many different formats and can even be stored as text files. These are the final commodity of ML training. Accordingly, the last step is to unit test the model.

Testing the model involves the following steps: use the model to make predictions for the test dataset, compare the predictions to the true values or labels by calculating various metrics, and evaluate the performance of the model. Assessing the model with the test dataset provides information about how well the model's predictions generalize to new data. The metrics can be used to evaluate the performance of the model and even allow for comparison between different ML algorithms.

There are many performance metrics that can be evaluated on the test dataset. In ML, precision (i.e., positive predictive value) and recall (i.e., sensitivity) are commonly used for classification problems. These values are commonly represented as the precision-recall (PR) curve, which is analogous in concept to the receiver operator characteristic (ROC) curve and can be similarly summarized by calculating the area under the PR curve (PRAUC).

In this experiment, evaluation of the model on the test dataset demonstrated a binomial classification accuracy of 93.12% (SCF Code Block 7) and a PRAUC of 0.97 (SCF Code Block 8). These results are similar to those presented by Wilkes et al. for binomial classification using XGBT with no subsampling or feature selection.

Discussion

In this article, we describe high-level principles of ML, using the development of a binary classification model as a contextual framework. As described above, the major themes encountered in the train-test development cycle include, (1) the appropriate separation of datasets, (2) the mechanics of ML training, and (3) model testing. While PAA profile classification is a specific and narrow use case, the workflow and underlying theory are broadly generalizable to modern applications of supervised ML.

Ensuring train and test data are truly separated is an essential consideration in ML development. Similar to contamination in NAAT testing, train-test contamination can happen inadvertently, and this so-called data leakage is not always easy to detect. For example, if two healthcare entities share electronic health information and a researcher unknowingly combines two separate extracts of laboratory result data from each of those institutions, duplicate entries will exist and could be represented in train and test datasets. Using an alternative training protocol, we illustrate the effect of train-test contamination by adding

test data to the training dataset at 10% intervals. Following this procedure, the binomial classification accuracy increases linearly as the amount of train-test contamination increases. With 100% train-test contamination (i.e., all test data is analyzed during training), evaluation of the resulting model demonstrated a binomial classification accuracy of 99% and a PRAUC of 0.999 (Supplemental Table). From this, we can observe that the failure to separate train and test data will result in inflated performance metrics with model testing. Robust quality assurance practices during development can offer reassuring data that train-test contamination does not exist but, these practices are not always implemented. Alternatively, robust validation with multiple hold-out datasets can also detect this type of error as one would expect significant performance degradation between evaluations with the contaminated and uncontaminated test datasets.

Lastly, the iterative nature of training and the optimization of a loss function remain paramount in the intuitive understanding of ML. In an alternative training protocol, the XGBT model was trained for 5 iterations. This resulted in a binomial classification accuracy of 87.5% and a PRAUC of 0.889. Knowing that our model is capable of higher performance metrics, this model would be considered 'underfit', in that, the model will perform better if it was given additional time to learn. However, if the algorithm is given too much time to learn, or is allowed to become too complex, the model may become overfit. Following an alternative training protocol where the algorithm is tuned towards high complexity, training loss is observed to decrease across all training iterations. However, the validation loss is observed to follow an initial decrease, followed by a steady increase (Supplemental Fig. 1). This would indicate a model is likely overfit. Evaluation of this model on the testing dataset did not appear to have deleterious effects on performance, indicating that data distributions between the train and test may be sufficiently similar. However, it would be expected that a model with evidence of overfitting would be unlikely to perform well with novel real-world data.

Several aspects of ML development, such as feature engineering and model interpretation, were omitted since detailed discussion of these steps are outside the scope of this paper but are important to briefly mention here. Feature engineering is the process of determining a set of features or inputs for the ML algorithm to train on and involves feature selection and extraction. Feature selection is the step of choosing the most useful inputs in the dataset to train on, and feature extraction is the process of making more useful inputs by combining existing features. Choosing a useful set of features is essential in successfully training a ML model. Model interpretability is also an important issue to note since many of the best performing ML models are often "black boxes." In our discussion, XGBoost provided predictions with high accuracy but is opaque to what and how it learned from the data. This contrasts with a model like linear regression, which has straightforward interpretations of its structure. Tools such as LIME (Local Interpretable Model-agnostic Explanations) and SHAP (Shapley Additive Explanations) can aid in providing insights in black box models, and balancing interpretability and predictive power must be considered in any ML development project [28,29].

Conclusion

The use of ML in the clinical MS laboratory offers the potential for several operational enhancements. However, there are several inherent limitations of current ML technology that can negatively affect the robustness of these applications. Several regulatory agencies are beginning to offer draft guidance on how ML can be safely implemented in clinical practice [30–33]. Regulations can be helpful, but strong technological literacy among laboratorians is also necessary to ensure these novel methods enhance our workflows and do not negatively impact the high-quality nature of test results. The concepts described here are fundamental and generalizable to supervised ML development and may provide an intuitive understanding of these novel technologies. Continued review of ML topics will be needed as the fields of artificial intelligence and computer science continue to evolve rapidly.

Disclaimers

None

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to acknowledge Rachel Carling and Gary Woodward et al., for their original work and their successful efforts in curating the plasma amino acid profile dataset. Thank you for your collaborative nature in the preparation of this manuscript, all in the spirit of education.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.jmsacl.2021.12.001.

References

[1] O. Ardon, R.L. Schmidt, Clinical laboratory employees' attitudes toward artificial intelligence, Lab Med. 51 (6) (2020 Nov 2) 649–654.
[2] K.P. Murphy, Machine learning: a probabilistic perspective, MIT Press, 2012.
[3] J.H. Harrison, J.R. Gilbertson, M.G. Hanna, N.H. Olson, J.N. Seheult, J.M. Sorace, et al., Introduction to artificial intelligence and machine learning for pathology, Arch. Pathol. Lab. Med. (2021 Jan 25).
[4] H.H. Rashidi, N.K. Tran, E.V. Betts, L.P. Howell, R. Green, Artificial intelligence and machine learning in pathology: the present landscape of supervised methods, Acad. Pathol. 6 (2019 Dec).
[5] B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, et al., mlr: Machine Learning in R, J. Mach. Learn. Res. 17 (1) (2016) 5938–5942.
[6] H. Wickham, M. Averick, J. Bryan, W. Chang, L. McGowan, R. François, et al., Welcome to the tidyverse, JOSS 4 (43) (2019 Nov 21) 1686.
[7] Team RC. R: A language and environment for statistical computing. 2013.
[8] T.E. Oliphant, Python for Scientific Computing, Comput. Sci. Eng. 9 (3) (2007) 10–20.
[9] Running the R Markdown Notebook [Internet]. [cited 2021 Nov 24]. Available from: https://github.com/edwardslee/R_paa_profile_classification.
[10] Python PAA Profile ML Walkthrough [Internet]. [cited 2021 Nov 24]. Available from: https://github.com/tjdurant/python_paa_profile_classification.
[11] E.H. Wilkes, E. Emmett, L. Beltran, G.M. Woodward, R.S. Carling, A machine learning approach for the automated interpretation of plasma amino acid profiles, Clin. Chem. 66 (9) (2020 Sep 1) 1210–1218.
[12] A.P.R. Zabell, T. Foxworthy, K.N. Eaton, R.K. Julian, Diagnostic application of the exponentially modified Gaussian model for peak quality and quantitation in high-throughput liquid chromatography-tandem mass spectrometry, J. Chromatogr. A 21 (1369) (2014 Nov) 92–97.
[13] F.B. Vicente, D.C. Lin, S. Haymond, Automation of chromatographic peak review and order to result data transfer in a clinical mass spectrometry laboratory, Clin. Chim. Acta 498 (2019 Nov) 84–89.
[14] Wang H, Wang H, Zhang J, Li X, Sun C, Zhang Y. Using machine learning to develop an autoverification system in a clinical biochemistry laboratory. Clinical Chemistry and Laboratory Medicine (CCLM). 2020 Nov 26;0(0).
[15] M. Yu, L.A.L. Bazydlo, D.E. Bruns, J.H. Harrison, Streamlining quality review of mass spectrometry data in the clinical laboratory by use of machine learning, Arch. Pathol. Lab. Med. 143 (8) (2019 Feb 20) 990–998.
[16] E.H. Wilkes, G. Rumsby, G.M. Woodward, Using machine learning to aid the interpretation of urine steroid profiles, Clin. Chem. 64 (11) (2018 Aug 10) 1586–1595.
[17] R. Arnaout, Machine learning in clinical pathology: seeing the forest for the trees, Clin. Chem. 64 (11) (2018 Sep 20) 1553–1554.
[18] J. Zhang, J. Rector, J.Q. Lin, J.H. Young, M. Sans, N. Katta, et al., Nondestructive tissue analysis for ex vivo and in vivo cancer diagnosis using a handheld mass spectrometry system, Sci. Transl. Med. (2017). Sep 6;9(406).
[19] Balog J, Sasi-Szabó L, Kinross J, Lewis MR, Muirhead LJ, Veselkov K, et al. Intraoperative tissue identification using rapid evaporative ionization mass spectrometry. Sci. Transl. Med. 2013 Jul 17;5(194):194ra93.

[20] I.R. White, J.B. Carlin, Bias and efficiency of multiple imputation compared with complete-case analysis for missing covariate values, Stat. Med. 29 (28) (2010 Dec 10) 2920–2931.
[21] Z. Che, S. Purushotham, K. Cho, D. Sontag, Y. Liu, Recurrent Neural Networks for Multivariate Time Series with Missing Values, Sci. Rep. 8 (1) (2018 Apr 17) 6085.
[22] A. Géron, Hands-on machine learning with Scikit-Learn. Keras and TensorFlow: Concepts, tools, and techniques to build intelligent systems, O'Reilly Media, 2019.
[23] R. Caruana, A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms. Proceedings of the 23rd international conference on Machine learning' ' - ICML '06, ACM Press, New York, New York, USA, 2006, pp. 161–168.
[24] T. Chen, C. Guestrin, XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD' ' '16, ACM Press, New York, New York, USA, 2016, pp. 785–794.
[25] Introduction to Boosted Trees — xgboost 1.5.0-dev documentation [Internet]. [cited 2021 Jul 10]. Available from: https://xgboost.readthedocs.io/en/latest/tutorials/model.html.
[26] S.N. Murphy, M.E. Mendis, D.A. Berkowitz, I. Kohane, H.C. Chueh, Integration of clinical and genetic data in the i2b2 architecture, AMIA Annu. Symp. Proc. 1040 (2006).
[27] Friedman J, Hastie T, Tibshirani R. Showing results for the elements of statistical learning. The elements of statistical learning. 1(10).
[28] Lundberg SM, Lee SI. A unified approach to interpreting model predictions. Proceedings of the 31st international conference on neural information processing systems. 2017;4768.
[29] M. Ribeiro, S. Singh, C. Guestrin, in: why should I trust you?": explaining the predictions of any classifier, Association for Computational Linguistics, Stroudsburg, PA, USA, 2016, pp. 97–101.
[30] Food and Drug Administration. Clinical and Patient Decision Support Software: Draft Guidance for Industry and Food and Drug Administration Staff [Internet]. 2017 [cited 2019 Jan 21]. Available from: https://www.fda.gov/downloads/medicaldevices/deviceregulationandguidance/guidancedocuments/ucm587819.pdf.
[31] Food and Drug Administration. Clinical Decision Support Software: Draft Guidance for Industry and Food and Drug Administration Staff' '. 2019 Sep 27.
[32] W.L. Schulz, T.J.S. Durant, H.M. Krumholz, Validation and regulation of clinical artificial intelligence, Clin Chem. 65 (10) (2019 Oct 1) 1336–1337.
[33] Artificial Intelligence and Machine Learning in Software as a Medical Device | FDA [Internet]. [cited 2021 Jul 15]. Available from: https://www.fda.gov/medical-devices/software-medical-device-samd/artificial-intelligence-and-machine-learning-software-medical-device.