Machine Learning Based Suggestions of Separation Units for Process Synthesis in Process Simulation

Jonas Oeing*, Fabian Henke, and Norbert Kockmann

DOI: 10.1002/cite.202100082

As part of Industry 4.0, workflows in the process industry are becoming increasingly digitalized. In this context, artificial intelligence (AI) methods are also finding their way into the process development. In this communication, machine learning (ML) algorithms are used to suggest suitable separation units based on simulated process streams. Simulations that have been performed earlier are used as training data and the information is learned by machine learning models implemented in Python. The trained models show good, reliable results and are connected to a process simulator using a .NET framework. For further optimization, a concept for the implementation of user feedback will be assigned. The results will provide the fundamental basis for future AI-based recommendation systems.

Keywords: Machine learning, Process engineering, Process simulation, Process synthesis, Separation units

Received: May 28, 2021; *revised:* July 14, 2021; *accepted:* September 08, 2021

1 Introduction

The chemical industry is subject to constant change due to global challenges and rapid product development. New challenges and needs require new products with related processes. The development of a digital twin [1] and the integration of artificial intelligence (AI) offer a multitude of new opportunities to reduce costs, time and engineering effort [2]. Particularly in the area of process development, which is predominantly based on heuristic decisions [3], data-driven models can find their application for engineering assistance. However, the increasing digitalization of the process industry continuously provides new data and opportunities, which can be used as a basis for AI or, in particular, machine learning (ML), which is able to learn rules and regularities from data by using statistical models. Thus, more and more AI solutions are finding application in the process industry. Asprion et al. developed a gray-box model based approach to optimize process simulations for units where rigorous models are missing [4]. In parallel, AI-based image recognition is being used from Schuler et al. to monitor multiphase flows [5] or classify crystals by Heisel et al. [6]. AI offers the opportunity to leverage the know-how that has been incorporated into process synthesis over the years and make it available in data-driven models.

In the following, it will be shown how a ML algorithm can be used to find the most suitable separation unit according to the prediction accuracy based on process values of material streams. Here, data from process simulations are used as training data and processed with various ML algorithms to investigate the potential. Subsequently, the models are linked in Python (Python Software Foundation) with the process simulation software CHEMCAD (Chemstations Europe GmbH) to prepare a direct prediction of separation units based on current simulation streams, resulting in a data-driven support of the process synthesis.

2 Synthesis and Training

The concept is based on a classical ML model, which consists of three parts (see Fig. 1). An input part containing a vector of process values (see Tab. 1), the actual ML model as a classifier and an output part, which gives the appropriate separation operation in the form of a class.

For this study, the most popular machine learning algorithms were used and compared against each other:
- K-Nearest-Neighbors [7, 8]
- Multinomial Logistic Regression [9]
- Naive-Bayes [10, 11]
- Decision Tree [12, 13]
- Random Forrest [14]
- Support Vector Machine (SVM) [15, 16]
- Artificial Neural Network (ANN) [17]
- One-vs-One [18]
- One-vs-All [19]

Jonas Oeing, Fabian Henke, Prof. Dr.-Ing. Norbert Kockmann
jonas.oeing@tu-dortmund.de
TU Dortmund University, Department of Biochemical and Chemical Engineering, Laboratory of Equipment Design, Emil-Figge-Straße 68, 44227 Dortmund, Germany.

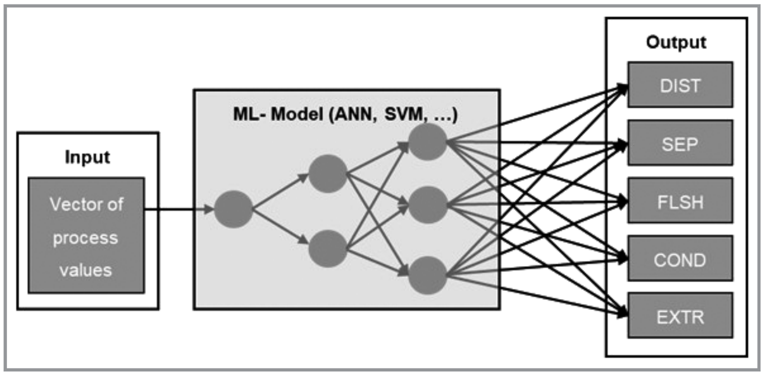

Figure 1. Structure of the machine learning approach for separation suggestions in process synthesis.

Table 1. Relevant substance properties and characteristic physical quantities for use as input parameters.

Substance properties [3]	Physical quantities
Molecular size	Gyration radius
Vapor pressure	Vapor pressure according to Antoine
Polarity	Dipole moment
Solubility	Solubility parameter according to Hildebrand
Melting point	Melting temperature
Boiling point	Boiling temperature

The implementation is done in Python (version 3.8.3) using the scikit-learn and keras libraries [20]. Since the present case has only a small amount of data for machine learning, training and test data will not be representative, resulting in a variance of the validation data. Cross-validation is used to minimize this variance. This allows for each class in the full data set to be represented in approximately correct proportions in the training and test sets. [21]

3 Database

The database is formed by example processes of the flowsheet simulator CHEMCAD. As input data, a vector of process values is used, which are taken from the respective process streams and databases stored in CHEMCAD. For initial validations, the focus is set on the separation of binary substance systems. In order to achieve an efficient result, it is important that the input data shows the necessary physical relationships, which are required for the selection of the separation. Typical separations are based, e.g., on the fact that at one point in the process the substances to be separated are present in different aggregate states. The melting temperature, the boiling temperature and the vapor pressure as a function of the process temperature are characteristic for describing such a separation. Separation processes such as extraction, which work with an additional solvent, are based on different solubility or polarity of the individual components. The size of a molecule can also be used for separation from a mixture of substances. Zeolites or other molecular sieves, for example, capture only small molecules, while larger particles remain in the bulk phase. These considerations result in the substance properties shown in Tab. 1, which have to be considered for the selection of a separation operation, as well as the respective characteristic physicochemical quantities. Goedecke et al. [3] list these material properties as relevant decision variables, too. The substance properties shown in the table represent all necessary physical relationships and are therefore extracted from databases stored in CHEMCAD for training. In case a parameter is not stored in the database, it is assigned the value 0. It is important to mention that for nearly all substances, only a few percent of the parameters (0–4 %) were not available in the databases. Only for the gyration radius and the Antoine parameters between 6–7 % of the data were not available, since these are not measured and stored for these substances, and were therefore replaced with 0 as a placeholder. In addition, the actual pressure and temperature of the process stream are considered as input values too, to integrate the current operation conditions.

The output is the corresponding separation unit, where units that require a further additive for separation (e.g., extraction, absorption) are combined into a *separation with additive* class. This results in the following classes considered in the data: phase separator, distillation, multipurpose-flash, condensation, separation with additive. The set of example simulations includes a total of 37 flowsheets.

From these simulations, the substances to be separated and their pure substance data were extracted for each individual separation unit and added to a training data set in tabular form. The substances were decomposed in such a way that binary separation problems are present and the pure substance data of the substances (see Tab. 1) represent the input data. Output data represent the associated separation unit.

4 Results

The models presented in Sect. 2 were validated and compared using a repeated stratified K-fold cross-validation with five folds and 100 repetitions. The decision to use five folds is justified by the fact that the class *condensation* has only 5 data sets available. In this way it should be possible that in each fold the class *condensation* is represented at least once.

4.1 Binary Substance Systems

The training scores obtained are shown in Fig. 2. The training score in this paper describes the mean percentage of correct suggestions over all repetitions of the cross-validation. It can be seen that all models except the Naive Bayes (49 %) and the Artificial Neural Network (45 %) are able to learn the relationships between the process values and the respective separation units. The best scores with over 90 % are provided by the Decision Tree and the Random Forest consisting of a large number of Decision trees. They are followed by Support Vector Machines (linear-kernel 85 %, poly-kernel 82 %, rbf-kernel 80 %) and the K-Nearest-Neighbors algorithm (83 %). One-vs-All and Logistic Regression have scores just below 80 %. For Naive Bayes, the low training score is probably due to the assumptions made by the algorithm. On the one hand, there are physically based correlations between some of the variables being considered. For example, the vapor pressure depends on the temperature of the stream. However, the Naive Bayes assumes (conditional) independence of the properties. Furthermore, the normal distribution of the data assumed by Naive Bayes does not apply to the individual properties. In addition, the standard deviation across all repetitions of the cross-validation is shown in Fig. 2. For all models, this is in similar ranges between four and six percentage points.

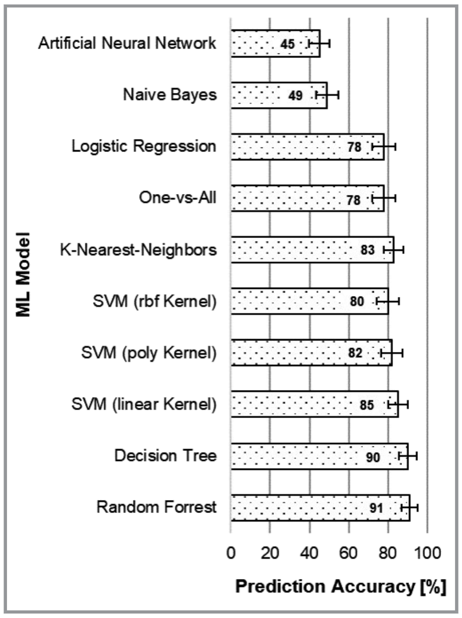

Figure 2. Comparison of the model performance (training scores, percentage of correct suggestions) based on a repeated stratified K-fold cross-validation.

In addition to the actual training score, there are other features that should be used to analyze the quality of the classification. For instance, the output of a confusion matrix provides information about how many results were correctly classified and in which situations a false positive or false negative classification took place. Fig. 3 shows the confusion matrix for a decision tree after training with a repeated stratified K-fold cross-validation. The entries represent the averaged absolute values for the test data over all folds and repetitions.

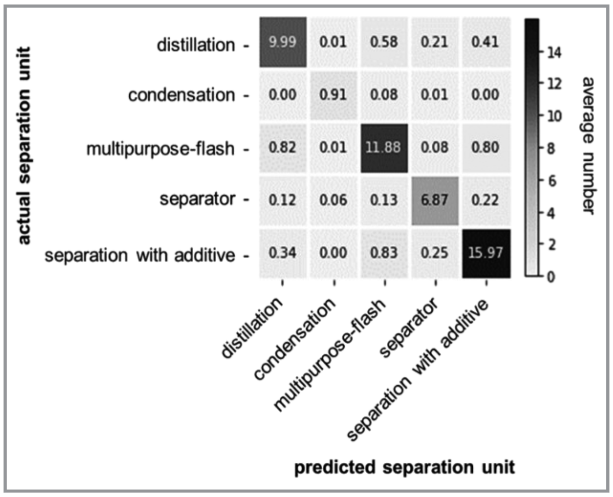

Figure 3. Average confusion matrix of classification results for a decision tree after cross-validation.

The actual separation units are placed vertically, while the predicted classes are horizontally arranged in Fig. 3. The diagonal of the matrix thus maps the correctly predicted classes. Assignments in fields outside the diagonal have thus been recognized incorrectly. A closer look at the matrix shows that almost all separation operations could be classified correctly. At the same time, it is noticeable that the data set has an uneven distribution of classes. Operations such as distillation, separation and separation with additive, which are more common in process plants, are clearly more prevalent in the test data than separation or condensation, which is due to the fact that the data set is a representative cross-section. The underrepresentation of certain classes affects training, so their influence needs to be further investigated.

The results presented show that the most reliable predictions are obtained using Decision Tree based models as well as Support Vector Machines. Consequently, these are used for further optimization and examined with respect to their optimization potential by adjusting hyperparameters. Hyperparameters are higher-level specifications of machine learning models, which show a high impact of its learning speed, results and complexity [22]. For this purpose, a grid search algorithm in combination with the previously described cross-validation is used. A grid is created containing discrete values for relevant hyperparameters. For the Decision Tree, as well as the Random Forest algorithm, three parameters were tuned. The *criterion*, which influences the quality of the split. The *minimal impurity decrease* that occurs when a node is split. The *minimal amount of* *samples* required at a leaf node. Additionally, for the random forest the number of trees will be considered in tuning. For the tuning of the SVMs with linear- and rbf-kernel the kernel coefficient *gamma* as well as the regularization parameter *C* will be changed over the grid. Also, for the SVM with the poly-kernel the *degree* of the polynomial kernel function and its independent term *coef0* will be considered. The tuning parameters used during the hyperparameter tuning are shown in Tab. 2. The values that provide the best performance in terms of the prediction accuracy are underlined. It can be seen that more often the outer parameters in the grid give the best results, so further consideration of values outside the chosen intervals could possibly give better results.

Based on this grid search, the algorithm trains and validates all possible combinations of hyperparameters and compares them against each other. In this way, the best hyperparameters among the grid explored were selected for each model [22]. The results of the tuning are shown in Fig. 4. The deviations of the score after hyperparameter tuning are shown in comparison to the scores before tuning.

It is obvious that the Support Vector Machines benefit from the tuning. In this way, the scores can be increased by up to 6.1 % (poly-kernel). At the same time, the optimization of the Decision Tree does not show a significant increase in the score. The Random Forrest algorithm even loses –1.2 % although the default parameters in the grid are assigned for tuning. The decrease of the score is caused by the fact that the training scores, due to the small data base, show a fluctuation of about 1 % despite cross-validation.

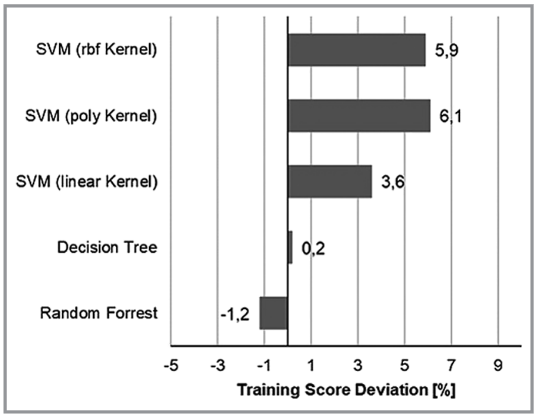

Figure 4. Deviation of training scores after hyperparameter tuning for different ML models.

4.2 Multi-substance Systems

Binary substance systems rarely occur in chemical industry. Because of this the ML models are tested on tertiary and quaternary substance systems, too. The necessary data is generated from the example processes mentioned in Sect. 3. For this purpose, tertiary and quaternary substance systems were derived from the previously used process streams and their pure substance data were defined as input and the associated separation unit as output. Fig. 5 shows the comparison of the performance of different ML models using binary, tertiary and quaternary systems. The tested models are

Table 2. Parameters used in hyperparameter tuning performed with grid search. The used algorithms are part of the python sklearn library. The underlined parameters provide the best results in terms of prediction accuracy.

	Hyperparameter	Tested values
Decision Tree [13]	criterion	gini, entropy
	min_impurity_decrease	0, 0.0000001, 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1
	min_samples_leaf	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Random Forest [14]	criterion	gini, entropy
	min_impurity_decrease	0, 0.0000001, 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1
	min_samples_leaf	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
	n_estimators	50, 100, 150
SVM linear [15]	C	0.01, 0.1, 1, 10
	gamma	auto, scale
SVM poly [15]	C	0.01, 0.1, 1, 10
	degree	1, 2, 3, 4
	gamma	auto, scale
	coef0	0, 1
SVM rbf [15]	C	0.01, 0.1, 1, 10
	gamma	auto, scale

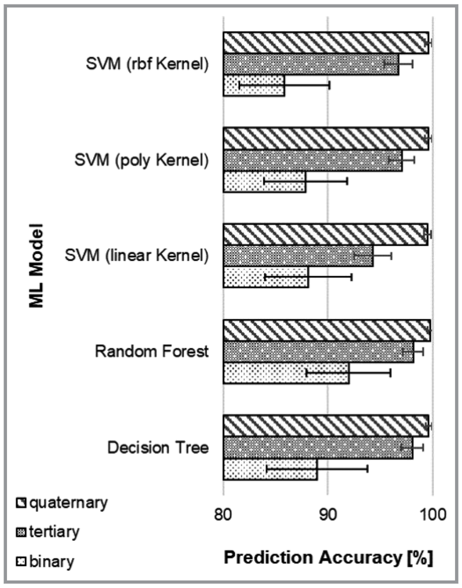

Figure 5. Comparison of the best model performance for multi-substance systems (training scores, percentage of correct suggestions) including standard deviation based on a repeated stratified K-fold cross-validation with tuned hyperparameters.

chosen based on the previous mentioned results of binary systems. For the quaternary substance systems, the class of *condensation* was not mentioned because of a lack of data.

The different ML models can learn coherences and generate suitable suggestions of separation operations for tertiary and quaternary substance systems. The scores reach high values between 94 % and 99 %. These high scores near to perfect suggestion accuracy (100 %) can be an indication for a possible overfitting. In this case, the ML models learn the coherences of the used data quite properly and often generate matching suggestions for the test data, because the different records are similar to each other. The algorithms reach high scores for data that is quite like the used training data. In the considered case, the test data fulfills this criterion because of the way the data is generated at the beginning and then used during the repeated stratified K-fold cross-validation. Based on this assumption, the scores for tertiary and quaternary substance systems must be interpreted as unrealistic high. Nevertheless, these ML models have a high potential to be used for multi-substance systems, too. The results of the presented studies show that the Decision Tree, Random Forest and SVM models are able to learn the relationships present in the data for all used datasets. However, since overfitting occurs for tertiary and quaternary substance systems, they need to be further analyzed and adapted in the future.

5 Feedback Optimization

The available training data set is quite small and non-uniformly distributed. For example, the class of condensation is underrepresented, and reliable results cannot be guaranteed even on new input data that deviate significantly from the training data due to their small amount. For this reason, a constant adaptation and improvement of the algorithm, respectively of the ML model, is performed during the usage. Therefore, a Python algorithm is developed, which is adapted by interaction with the user, similar to reinforcement learning [23]. The principle of this program is shown in Fig. 6.

The main component of the interactive algorithm is a ML model trained according to Sect. 2, which receives the data of a stream from a process simulator. The model calculates the most suitable separation unit based on the relationships

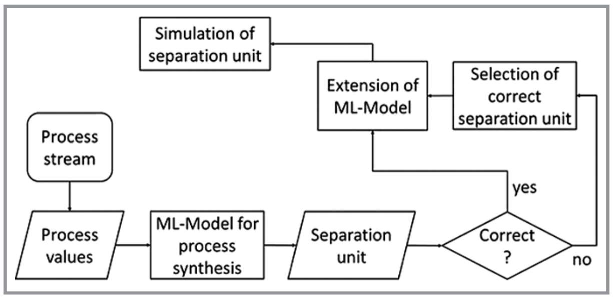
Simulation of separation unit
Extension of ML-Model
Selection of correct separation unit
Process stream
yes
Process values
ML-Model for process synthesis
Separation unit
Correct ?
no

Figure 6. Structure of an interactive script for user feedback optimization.

learned from the training data and could suggest also second and third ranked separation processes. Afterwards, the user decides whether the suggested separation unit is suitable. If this is the case, the separation unit is simulated, and the output stream determined in the simulation can be used in a next step as a new input for the ML model and the program runs again. In this way, sequential design of separation processes can be performed with the help of the ML model. In parallel, the input values and the results are appended to the training data set and the model is trained again to further improve the prediction accuracy. If the prediction is wrong, the user selects the most suitable separation unit by a manual input and the training data set is extended by the input vector and the selected unit. Subsequently, the extension is used to train the model, which results in an optimized and robust model. At the same time, the extension allows the model to continue learning with each new application and to be able to cover new use cases.

Initial feasibility studies in our group show a reliable extension of the data set with further adjustments to increase efficiency in the future. For example, care must be taken to ensure that the expansion of the training data does not result in too large and uneven distribution within the data set. Correction factors may have to be inserted to give more weight to underrepresented classes.

6 Implementation in Process Simulation

As shown in Fig. 7 a link to CHEMCAD via the CC-API .NET Python interface [24] is implemented into the feedback optimization as well. This amplifies the using options of the algorithm. The user must hand over an existing CHEMCAD flowsheet and an appropriate internal stream number to the script. Based on the information stored in CHEMCAD the algorithm calls up the necessary data and generates a suggestion of a separation operation for the chosen stream. The feedback optimization process is analogous to the representation in Fig. 6.

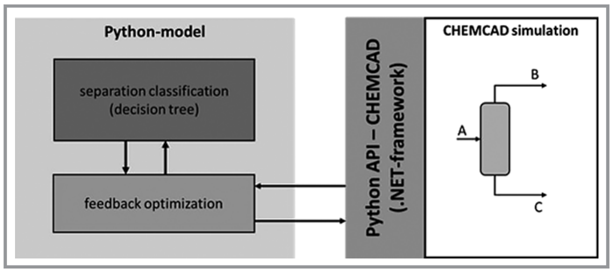
Python-model
separation classification (decision tree)
feedback optimization
Python API – CHEMCAD (.NET-framework)
CHEMCAD simulation
A
B
C

Figure 7. Connection of the Python-model in CHEMCAD via .NET Python API.

7 Conclusion and Outlook

With increasing digitalization, data is becoming more and more important in the process industry. As a result, more and more data can be made available and the know-how contained therein can be extracted for further use. In this paper, an ML approach for data-driven process development was demonstrated to predict suitable separation units based on substance properties of the material streams. First validations show promising results. Thus, separation units can be predicted with an accuracy of more than 90 % using the developed approach. Not only binary systems were considered, but also multi-substance systems and their operation were verified. In addition, an optimization was implemented that continuously extends the model with the help of user feedback. This allows an increase of the robustness and optimization during the application to substance systems that strongly deviate from the training data set. In addition to pure optimization, it is also interesting to investigate which parameters are particularly strongly involved in the decision-making process by the ML model. These should be given more attention in the future and their values should be experimentally determined if information is missing in databases.

Regarding future applications, it is important to keep in mind that trustworthy results will only be achieved if the previously human-generated data is reliable. For this reason, the training data must be verified and provided with metadata or a quality code that provides information about the trustworthiness of the data. The same applies to the user feedback and the resulting optimization. The need for approaches to reliable data storage is also propagated outside the KEEN project by the *German Catalysis Society* (GeCatS) [25].

Therefore, the developed approach is a recommending system, which supports the user during the process development. The decision whether a separation unit is suitable or not is thus still made by the user, which means that the question of the trustworthiness of the training data can play a subsidiary role. If the explainability of the results can be ensured by verified and reliable data, it is conceivable to extend the approach so that an autonomous synthesis of separation sequences takes place. However, then the question arises, who is responsible for the decisions from the program for a non-successful plant installation.

The BMWi is acknowledged for funding this KEEN project initiative (Support code: 01MK20014S). Open access funding enabled and organized by Projekt DEAL.

Abbreviations

AI	Artificial intelligence
ML	Machine learning
SVM	Support vector machine

References

[1] A. Bamberg, L. Urbas, S. Bröcker, N. Kockmann, M. Bortz, *Chem. Ing. Tech.* **2020**, *92 (3)*, 192–198. DOI: https://doi.org/10.1002/cite.201900168
[2] *Working Paper: Artificial Intelligence in Industrie 4.0*, Bundesministerium für Wirtschaft und Energie (BMWi), Berlin **2019**.
[3] R. Goedecke, W. Hofen, R. Sass, H. Wendeler, G. Schembecker, G. Wozny, H. Hahn, W. Albert, in *Fluidverfahrenstechnik* (Ed: R. Goedecke), Wiley-VCH Verlag, Weinheim **2006**.
[4] N. Asprion, R. Böttcher, R. Pack, M.-E. Stavrou, J. Höller, J. Schwientek, M. Bortz, *Chem. Ing. Tech.* **2019**, *91 (3)*, 305–313. DOI: https://doi.org/10.1002/cite.201800086
[5] J. Schuler, L. M. Neuendorf, K. Petersen, N. Kockmann, *AIChE J.* **2021**, *67 (2)*, e17111. DOI: https://doi.org/10.1002/aic.17111
[6] S. Heisel, J. Ernst, A. Emshoff, G. Schembecker, K. Wohlgemuth, *Powder Technol.* **2019**, *345*, 425–437. DOI: https://doi.org/10.1016/j.powtec.2019.01.018
[7] https://scikit-learn.org/stable/modules/neighbors.html#classification (accessed on May 10, 2021)
[8] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html (accessed on May 10, 2021)
[9] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (accessed on May 10, 2021)
[10] https://scikit-learn.org/stable/modules/naive_bayes.html (accessed on May 10, 2021)
[11] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB (accessed on May 10, 2021)

[12] https://scikit-learn.org/stable/modules/tree.html (accessed on May 10, 2021)
[13] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier (accessed on May 10, 2021)
[14] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier (accessed on May 10, 2021)
[15] https://scikit-learn.org/stable/modules/svm.html#svm-classification (accessed on May 10, 2021)
[16] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (accessed on May 10, 2021)
[17] N. Ketkar, in *Deep learning with Python: A hands-on introduction* (Ed: N. Ketkar), Apress. Berkeley, CA **2017**.
[18] https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsOneClassifier.html (accessed on May 10, 2021)
[19] https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html (accessed on May 10, 2021)
[20] A. Géron, K. Rother, T. Demmig, *Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme*, 2nd ed., O'Reilly Verlag, Sebastopol, CA **2020**.
[21] I. H. Witten, E. Frank, M. A. Hall, *Data mining: Practical machine learning tools and techniques*, The Morgan Kaufmann series in data management systems, Morgan Kaufmann, Amsterdam **2011**.
[22] D. Chicco, *BioData Min.* **2017**, *10*, 35. DOI: https://doi.org/10.1186/s13040-017-0155-3
[23] G. Rebala, A. Ravi, S. Churiwala, *An introduction to machine learning*, Springer, Berlin **2019**.
[24] A. Fricke, J. C. Schöneberger, in *27th European Symposium on Computer Aided Process Engineering*, Vol. 40, Computer Aided Chemical Engineering, Elsevier, Amsterdam **2017**.
[25] *White Paper: The Digitalization of Catalysis-Related Sciences*, German Catalysis Society, Frankfurt **2019**.