Neunzig, Christian; Möllensiep, Dennis; Kuhlenkötter, Bernd; Möller, Matthias

ML Pro: digital assistance system for interactive machine learning in production

Journal of Intelligent Manufacturing

Provided in Cooperation with:
Springer Nature

Suggested Citation: Neunzig, Christian; Möllensiep, Dennis; Kuhlenkötter, Bernd; Möller, Matthias (2023) : ML Pro: digital assistance system for interactive machine learning in production, Journal of Intelligent Manufacturing, ISSN 1572-8145, Springer US, New York, NY, Vol. 35, Iss. 7, pp. 3479-3499,
https://doi.org/10.1007/s10845-023-02214-0

This Version is available at:
https://hdl.handle.net/10419/317749

Journal of Intelligent Manufacturing (2024) 35:3479–3499
https://doi.org/10.1007/s10845-023-02214-0

ML Pro: digital assistance system for interactive machine learning in production

Christian Neunzig[1,2] · Dennis Möllensiep[1] · Bernd Kuhlenkötter[1] · Matthias Möller[2]

Received: 18 April 2023 / Accepted: 5 September 2023 / Published online: 4 October 2023

Abstract
The application of machine learning promises great growth potential for industrial production. The development process of a machine learning solution for industrial use cases requires multi-layered, sophisticated decision-making processes along the pipeline that can only be accomplished by subject matter experts with knowledge of statistical mathematics, coding, and engineering process knowledge. By having humans and computers work together in a digital assistance system, the special characteristics of human and artificial intelligence can be used synergistically. This paper presents the development of a digital human-centered assistance system for employees in the production and development departments of industrial manufacturing companies. This assistance system enables users to apply production-specific data mining and machine learning techniques without programming to typical tabular production data, which is often inherently high-dimensional, nonstationary, and highly imbalanced data streams. Through tight interactive process guidance that considers the dependencies between machine learning process modules, users are empowered to build and optimize predictive models. Compared to existing commercial and academic tools with similar objectives, the digital assistance system offers the added value that both classical shallow and deep learning as well as generative and oversampling methods can be interactively applied to all feature table use cases for different user modes without programming.

Keywords Failure prognosis · Predictive quality control · Supervised learning · Human–machine interaction · Interactive machine learning

Abbreviations

AI	Artificial intelligence
API	Application programming interface
ANN	Artificial neural networks
AutoML	Automated machine learning
BA	Business application
CASH	Combined algorithm selection and hyperparameter optimization
CLI	Command-line interface
CML	Classical machine learning
CV	Cross-validation
DAS	Digital assistance system
DM	Data mining
DS	Data science
EML	Explainable machine learning
GUI	Graphical user interface
HP	Hyperparameter
HPO	Hyperparameter optimization
AI	Artificial intelligence
API	Application programming interface
ANN	Artificial neural networks
AutoML	Automated machine learning
BA	Business application

✉ Christian Neunzig
christian.neunzig@boschrexroth.de;
neunzig@lps.ruhr-uni-bochum.de

Dennis Möllensiep
moellensiep@lps.ruhr-uni-bochum.de

Bernd Kuhlenkötter
kuhlenkoetter@lps.ruhr-uni-bochum.de

Matthias Möller
matthias.moeller@boschrexroth.de

[1] Lehrstuhl Für Produktionssysteme, Ruhr-Universität Bochum, Bochum, Germany

[2] Bosch Rexroth AG, Homburg, Germany

CASH	Combined algorithm selection and hyperparameter optimization
CLI	Command-line interface
CML	Classical machine learning
CV	Cross-validation
DAS	Digital assistance system
DM	Data mining
DS	Data science
EML	Explainable machine learning
GUI	Graphical user interface
HP	Hyperparameter
HPO	Hyperparameter optimization
ICA	Independant component analysis
IDE	Integrated development environment
IML	Interactive machine learning
IT	Information technology
JSON	JavaScript Object Notation
LDA	Linear discriminant analysis
LR	Linear regression
LSR	Lasso regression
LGBM	Light gradient boosting machine
ML	Machine learning
RFR	Random forest regressor
RR	Ridge regression
SVR	Support vector regression
UCD	User-centered design
XGB	Extreme gradient boosting

Introduction

Recent research in applied machine learning in manufacturing such as that of Kaji et al. (2023), Na and Yang (2023) are concerned with solving a specific application. The demand for specialists who can competently deal with information technology issues and data-driven analysis methods is increasing worldwide (Zhang et al., 2022). Data preparation using data mining (DM) and data science (DS) methods are essential for the successful application of machine learning (ML) techniques. Selecting the appropriate algorithm, constructing proper input features by experimenting with different parameters, and testing are challenging for non-experts (Filz et al., 2023). Most non-experts do not know which machine learning algorithms work optimally before trying them. Even experts often apply different ML algorithms when they deal with data that is not common in ML (Lee, K., Yoo, J. et al. 2019). Especially for the application of ML in the manufacturing industry, besides the information technology (IT) and programming knowledge as well as the mathematical-statistical knowledge, the process knowledge from the respective application area is of great importance in order to be able to use the full potential of the available database (Dehnbostel et al., 2021; Xames et al., 2023), see (Pokorni et al., 2021). In the future, a specialized AI-engineer will be needed to cover the extensive spectrum of competencies (Krüger et al., 2019). Wider acceptance of ML can be achieved, as with other computer technologies that have a large user base (e.g., email interfaces, web technologies), by improving the usability of tools for use. Due to the digitalization of the industrial working world and demographic change, digital assistance systems (DAS) are gaining strong importance in various corporate sectors across industries to accelerate the qualification and induction of a large number of skilled workers for new technologies such as ML (Apt et al., 2018b).

DAS are all systems that support employees in their actions and are embedded in a higher-level IT system (Mättig & Kretschmer, 2019). To classify DAS, Apt et al. introduce different expressions for the degree, objective and type of assistance (Apt et al., 2018a). For the degree of assistance, the categories low, medium, high, and variable are introduced. A low degree of assistance is shown for simple assembly activities, for instance. Examples for medium degree of assistance are rule-based planning processes of medium complexity. A high level of assistance is required for complex rule-based decision-making processes and open-ended, creative problem-solving processes. The degree of assistance can be variable in team structures with different levels of training and complexity of requirements. The objective of the assistance ranges from compensatory, preserving to skill-enhancing (Apt et al., 2018b). Compensatory and sustaining assistance systems offer the opportunity to balance individual, activity-related deficits and to exploit inclusion potentials along the diversity dimensions of age, disability, and ethnicity (Apt et al., 2018a).

In terms of the type of assistance, physical assistance systems are distinguished from DAS as execution assistance systems (exoskeletons; human–robot collaborations) (Busse et al., 2020). DAS supports humans in task performance in the manner of sensory perceptual support or cognitive decision making. In practice, the simplest example is the selective provision of information such as hints in the user interface of software, on the other hand, multi-step decision-making processes can be supported as well (Link & Hamann, 2019). The benefit of DAS follows from its ability to perform intelligent procedures that humans are inferior or fail to perform in the face of high levels of difficulty and system complexity due to their limited cognitive abilities (Blutner et al., 2007).

Bartschat et al. divide the user groups for computer–human interaction in DM tools related to ML into four groups: business applications (BA), applied research, algorithm development, and education (Bartschat et al., 2019):

1. BA: This group uses DM and ML as tools to solve commercially relevant BA, such as process optimization and automation. BA are covered by commercial tools that provide support for big data databases and deep integration with enterprise workflows.
2. applied research: a user group that applies DM and ML to research problems. Here, users are primarily interested in tools with proven methods, a GUI, and interfaces to domain-specific data formats or databases.
3. algorithm development: these users develop new ML algorithms and need tools to integrate their own methods as well as to compare them with competitive algorithms.
4. Education: for university education, ML and DM tools should be intuitive, have a convenient interactive user interface, and should be affordable.

In the context of this research work, a human-centered DAS was developed, which enables employees of the development and planning departments of industrial companies to apply advanced methods of DM, DS and especially ML to tabular data sets without developing lines of code. These employees generally bring mathematical-statistical as well as process knowledge from their university education, but due to shifting emphasis to IT and ML, they are usually well-versed with basic knowledge in the new areas (Apt et al., 2018b). Moreover, many users of the existing tools still fail to use them due to their complicated application. Even the operation of these tools requires extensive training. In addition, a selection of methods is provided without a suitability check whether all necessary domain requirements are covered. A DAS should provide users with step-by-step guidance and support for numerous decisions along the development process of an ML solution, with the goal of developing ML solutions in production within applied research. Typical challenges of real-world production data are nonstationary, high-dimensional datasets with unbalanced class ratios of good to bad parts. In ML Pro, a simple incorporation of typical coping strategies for the challenges in production datasets is provided for non-experts. In this work, the research question is investigated:

How must a digital assistance system be designed so that employees of development and planning departments of industrial companies can apply production-typical ML methods to tabular data sets without programming?

In this article, related work on existing solution approaches and requirements for DAS to apply ML (Chapter 2) is discussed. Subsequently, the structure, design characteristics, and utility of the developed DAS *ML Pro* are presented (Chapter 3) by demonstrating the functionality of *ML Pro* on a real-world, open-access production dataset. Chapter 4 discusses the contributions of *ML Pro* to research. Further research potential and key messages are noted as well.

Human–machine interaction in the implementation of machine learning techniques

The modes of interaction between humans and computers for the application of DM, which are transferable to the application of ML, can be divided into the following different categories (Bartschat et al., 2019):

1. Text-only interface with a programming language (command-line interface, CLI)—difficult to handle, but easy to automate,
2. Integrated development environment (IDE)—easy to use, but difficult to automate,
3. Graphical user interface (GUI); where the user selects function blocks or algorithms from a collection, defines parameters, places them in a workspace and connects them together to create complete ML solutions—a good compromise, but difficult to handle for large workflows,
4. Visualization interfaces such as dashboards with a collection of different charts to display different types of data and results.

Csiszar et al. emphasize the importance of a user-centered design (UCD) structure for communicating with ML solution generation tools for users with domain knowledge. The user group in manufacturing use applied research to develop ML applications. For domain experts to trust and accept newly introduced AI or ML applications, the integration of UCD into the ML development process and the incorporation of insights from interactive and interpretable ML research are recommended (Csiszar et al., 2020). An interactive system is the combination of humans, hardware, and software with which users interact to achieve specific goals (DIN 9241). The selected overall framework for computer–human interaction for applying ML using DAS is interactive machine learning (IML). In this chapter, IML and competing DAS based on IML for applying ML are discussed.

Interactive machine learning

The communication model of IML centers the human as the decision maker ("human-in-the-loop"). IML differs from classical machine learning (CML) in that it uses human intelligence by iteratively teaching and refining the model in a relatively tight loop of computation and subsequent verification. This contrasts with CML, where the workflow requires extensive reselection of training data and significant changes to the model per execution step. Amershi et al. (2015) discuss the difference between IML and CML and present several case studies that highlight the value of this approach. In IML, a mathematical model is established and refined through iterative cycles of input and verification by a group of users. The

refinement of the model is done, for example, by providing and preparing selected data, applying the various operations for feature construction, or selecting algorithms as well as appropriate model parameters, so-called hyperparameters (HP), based on essential decisions made by the human, but prepared and executed by the machine in a supportive manner (Bernardo, 2020). Thus, it is possible for users to provide the system with additional information from the domain and process knowledge from the specific application. This does not mean that the computer has no influence on the development process or does not make independent decisions. For example, the computer can automatically select a subset of data and feed it into the algorithm. IML proposes to give users high-level control over the behavior of the system; without having to exercise this in every single interaction. From the interaction between computers and humans, the strengths of both interactors are synergized (Dudley & Kristensson, 2018).

Derivation of requirements for human–machine interaction

General requirements for the application of IML in a DAS are collected and derived. This is differentiated into tasks and competencies to be fulfilled by users (by Danyluk & Buck, 2019) and design rules (by Lee et al., 2019) for the development of such tools. The proposed framework for the platforms used to develop ML projects includes design rules that describe the functionalities with respect to task performance as well as design rules for the creation of the software are recommended, focusing on the characteristics and the structure.

General tasks of the users within a ML development process

Danyluk et al. elicited twelve essential tasks ((1)–(12)) required for the development process of an ML solution, presented in Table 1 (Danyluk & Buck, 2019):

Users' cognitive load can be reduced for task performance and decision making by providing unambiguous information in the proper place at an adequate level, as well as action strategies adapted to users' prior experiences (DIN 9241). Therefore, the essential decisions in development are addressed from problem description to model selection and configuration to result interpretation. The tasks and competencies of Danyluk et al. are also in line with more recent analyses of Rosemeyer et al (2022). which deal with the competence spectrum of ML from the perspective of shopfloor managers.

Table 1 Tasks and competences of ML-Platform users. (Danyluk & Buck, 2019)

1	Comparing broad classes of learning approaches, focusing on inputs and outputs to the algorithm, and ranges of problem types
2	Selecting and applying a broad range of ML tools to real-world data
3	Derive a learning algorithm from first principles and/or justify a learning algorithm from a mathematical, statistical, or IT perspective
4	Formally express representational power of models, critically interpret expressiveness & evaluate overfitting
5	Apply methods to mitigate overfitting and the "curse of dimensionality"
6	Provide an appropriate performance metric for evaluating ML algorithms
7	Application of an appropriate performance metric to evaluate ML algorithms
8	Applying appropriate evaluation methods to compare different ML algorithms
9	Implementing ML programs based on their algorithmic specifications
10	Become aware of bias in algorithms and data
11	Evaluate implications of decisions resulting from ML inferences
12	Compare differences in interpretability of learned models

Design rules of the ML software in respect of functionality and task performance

The task-related and usage order-related structuring refers to the development process of an ML pipeline (DIN 9241). The following twelve design rules regarding functionality for the ML development are proposed by Lee et al. shown in Table 2 (Lee et al., 2019).

Overall, the tasks and functions are listed in the order of the ML development process from data import to model creation to model adaptation to new data streams. Here, decisions must be made several times along the development process. A DAS supports the full decision process if the following three sub-processes are mapped:

(1) decision preparation,
(2) decision making as a choice between multiple alternatives, and
(3) decision execution.

Table 2 Functions and tasks of ML software according to (Lee et al., 2019)

1	Acquisition and segmentation of data
2	Pre-processing of data: Data cleaning and various functionalities for data transformation
3	Feature engineering: feature construction, feature selection, and feature extraction (by Lee et al. only "feature extraction"; extended here)
4	Definition of the ML task (regression, classification, etc.)
5	Selection of a ML algorithm
6	Configuration of ML algorithms
7	Possibility to measure model performance
8	Provision of computational resources
9	Implementation and application of created models in a productive environment
10	Ability to run ML algorithms (application & handling errors during training)
11	Ability to evaluate and compare different trained models & select the best model
12	Ability to update models to new data or changes in data streams

It follows that a DAS for decision support must be characterized by the following functions: Identification of a solution set, selection and evaluation of alternatives, and autonomous action (Blutner et al., 2007).

The data formats offered within DM and ML tools are based by the data structure of the raw data. Depending on the type of data and the goal of the DM process, the representation of the raw data is tailored to the task at hand. Moreover, the same raw data can be represented differently depending on the requirements of different tasks. The most widely used representation for a data set $\mathcal{D} = \left\{(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}\right\}$ with data points of features x_i and target variable y_i with $i \in \{1 \ldots N\}$ is the 2D feature table with N data point rows and d feature columns. Similarly, event logs and text data are two-dimensional, whereas time series data, data of sequences (genes, mass spectrogram), audio signals are 3D data. Video data and 3D images are 6D. 3D video files result in a 6D data base. The data formats of higher dimension than 2 can often be reduced to a 2D space by projection within the feature engineering (Bartschat et al., 2019).

Design rules of the ML software in respect of structure and characteristics

Lee et al. recommend general design rules for the software platform to perform ML, which are briefly introduced below (Lee et al., 2019). *User management* allows users to access their own projects. *Project management* allows users to create and manage multiple use cases in the DAS. An implemented resource management regulates the administration of created models and data in a folder system. *Data processing* is to be implemented to store and track the data processing steps that are executed on the models. The *model executor* is an implemented functionality that handles the creation of a model and the implementation of user inputs and configurations of the model and HP into the model. A *wrapper manager* is used to provide functionality from ML libraries such as sklearn or Tensorflow to users without having to program. The *model manager* maintains metadata on a project-specific basis for each model. This is information such as model name, creation time or specific information for training the model.

Overview of existing solutions of ML assistance systems for non-experts

This section provides an overview of existing solution building blocks and complete solutions of DAS to support non-experienced users and developers. The overarching logic is the division into generic tools for different domains and the specific tools for a solution. Many solutions converge supporting non-experienced ML users with automated ML (AutoML). With Vertex-AI, Google offers a commercial service with a web application for AutoML that allows developers with limited knowledge of ML to develop an ML model in the platform environment from import to output and interpretation of predictions for data in image, table, text, and video formats (Ng et al., 2022). Similar platforms are offered by Amazon, Microsoft, Apple, IBM, SAP, Tableau, MATLAB, H2O, SAS, and others. Prior research regarding AutoML has focused on the automatic determination of hyperparameters in machine algorithms. (Feurer et al., 2015; Moore, 2018; Thornton et al., 2012).

This work focuses on open-source solutions and libraries. This includes both application programming interface (API) and GUI. Several reviews have already addressed the solution of ML development in ML tools. Bartschat et al. give a descriptive and categorizing overview of commercial and academic solutions for AutoML, but do not bring any concrete design characteristics of an ideal tool for this purpose. Such a design is succinctly described in the comments of Lee et al. Bernardo shares design features for IML from the Rapid-Mix research project (Bernardo, 2020). Khalajzadeh et al. present an academic solution approach for cooperation and communication between domain experts and data specialists using the BiDaML tool to generate code templates based on diagrams to solve the use case (Khalajzadeh et al., 2020). Naik et al. compare well-known open-source tools with respect to the prediction accuracy of different algorithms for open-source datasets. No information is given about versioning or different implementation of the used algorithms in the tools (Naik & Samant, 2016). The same

problem occurs in ogunleye et al. where the self-developed solution only rudimentarily fulfills the functions of a complete ML pipeline (Ogunleye et al., 2019). Slater et al. studied 40 tools commonly used for DM in education. They highlight the utility of the Python programming language and the package Jupyter Notebooks, (Slater et al., 2017). Santos-Pereira et al. provide a survey of DM tools suitable for medical datasets. Medical datasets often exhibit characteristics that are also known in manufacturing datasets: high-dimensional, time-dependent, and non-stationary data streams, decentralized storage, different data formats (numeric, text, image, graph, audio, text, video, etc.), imbalanced data distributions (Santos-Pereira et al., 2022). The characteristics are confirmed by Dogan et al. without dealing in detail with the tools for solving the ML use cases in production (Dogan & Birant, 2021). Across the board, all reviews highlight the utility of open-source libraries such as R, scikit-learn, NumPy, SciPy, Pandas, PyCaret, etc. According to Bernardo, Google Tensorflow (Abadi et al., 2016), Apple CoreML1, TuriCreate2, and Pytorch3 are gaining acceptance as ML development toolkits and APIs for developing deep learning solutions (Bernardo, 2020). Whereas most of these APIs are aimed at ML experts, some of them are also aimed at ML non-experts. However, many of these APIs remain difficult to use. A survey of the software industry highlights impairing factors for users: lack of obvious benefits, ML development experience, and learning time to adopt ML tools (Spain, 2017). Well-known open-source tools that aim to reduce the learning time of using ML by non-experts by providing a GUI are WEKA (Witten et al., 2016), RapidMiner (Mierswa et al., 2006), KNIME (Berthold, M., R. et al. 2009) and Orange (Demsar et al., 2013) (Bartschat et al., 2019; Jovic et al., 2014). Shen and Sun introduce the notion of GraphicalAI, whose graphical programming implementation is also applied in the previously listed well-known tools. Ready-made code modules interlock with each other based on linked graphical blocks with specific ports (Shen & Sun, 2021).

From the reviews, the following conclusions can be drawn regarding the listed tools. The tools are offered with basic algorithms like linear regression, but also more complex algorithms like support vector machines or tree-based methods. Modern algorithms like extreme gradient boosting (XGB), light gradient boosting (LightGBM) or artificial neural networks (ANN) (Mishra & Srivastava, 2014) are not implemented and must be added either by compatible libraries or manually. KNIME is easy to learn and use due to the already implemented algorithms. Different data types can be processed. The steps of reading in data, preprocessing, cleaning, analysis, and output as well as different possibilities for visualization are available. The scope of data cleaning and analysis is like that of WEKA and RapidMiner. However, KNIME provides few methods for error measurement and no wrapper methods for descriptor selection. Strengths of WEKA are the ease of use of a large collection of methods for classification, clustering, and data collection in a GUI. However, the CLI is significantly more powerful than the GUI. In addition, no features can be developed. RapidMiner also has low capabilities in feature development. However, it offers CV capabilities and a variety of evaluation metrics compared to other tools. Documentation and high cost if used commercially are major drawbacks with RapidMiner. Orange offers great clarity compared to the other tools, but it is only designed for smaller data sets and thus not suitable for all production applications (Slater et al., 2017).

In addition to the described tools, which were developed generically for the widest possible range of uses, a selection of existing special solutions are listed, which were developed specifically for a particular use case: Avramidis developed a GUI for the evaluation of texts (Avramidis, 2017). Bahiuddin et al. present an ML application in a GUI for the study of magnetorheological fluids (Bahiuddin et al., 2017). Nasoz and Shrestha developed a web-based GUI for which three ML algorithms linear regression, logistic regression, and backpropagation are implemented for the classification task of breast cancer (Yamamoto 2017). Carney et al. also proposed a web application for classification of image data for the use in education (Carney et al., 2020). Amashaa et al. present a Python-based GUI for clustering algorithm application (Amashaa et al., 2020). Klemm et al. have developed a tool to develop, train and test an ANN using a GUI (Klemm et al., 2018). doty et al. present a draft GUI for few-shot learning for classification of electron microscopy image data (Doty et al., 2022). Milde et al. present a GUI for the development of Convolutional Neural Networks for the analysis of image data (Milde et al., 2018). Bashir et al. present a MATLAB-based tool for outlier detection of 3D time series data from sensors (Bashir et al., 2022). Another MATLAB-based GUI for the application of multiblock data analysis of up to four sensors is proposed by Mishra et al. (Mishra et al., 2020).

Moreover, helpful solution modules are continuously published within the ML pipeline for open-source application on relevant platforms such as Kaggle or Github, which have not yet made their way into the tools offered. Zöller et al. highlight the importance for explainability of AutoML applications and develop a visualization tool to increase user confidence through explainable ML (Zöller et al., 2022). Another solution module for explainable ML is the API Shapash, which allows ML users to get an explanation for a single (local) prediction of an algorithm and thus contributes significantly to understanding within the ML development process.

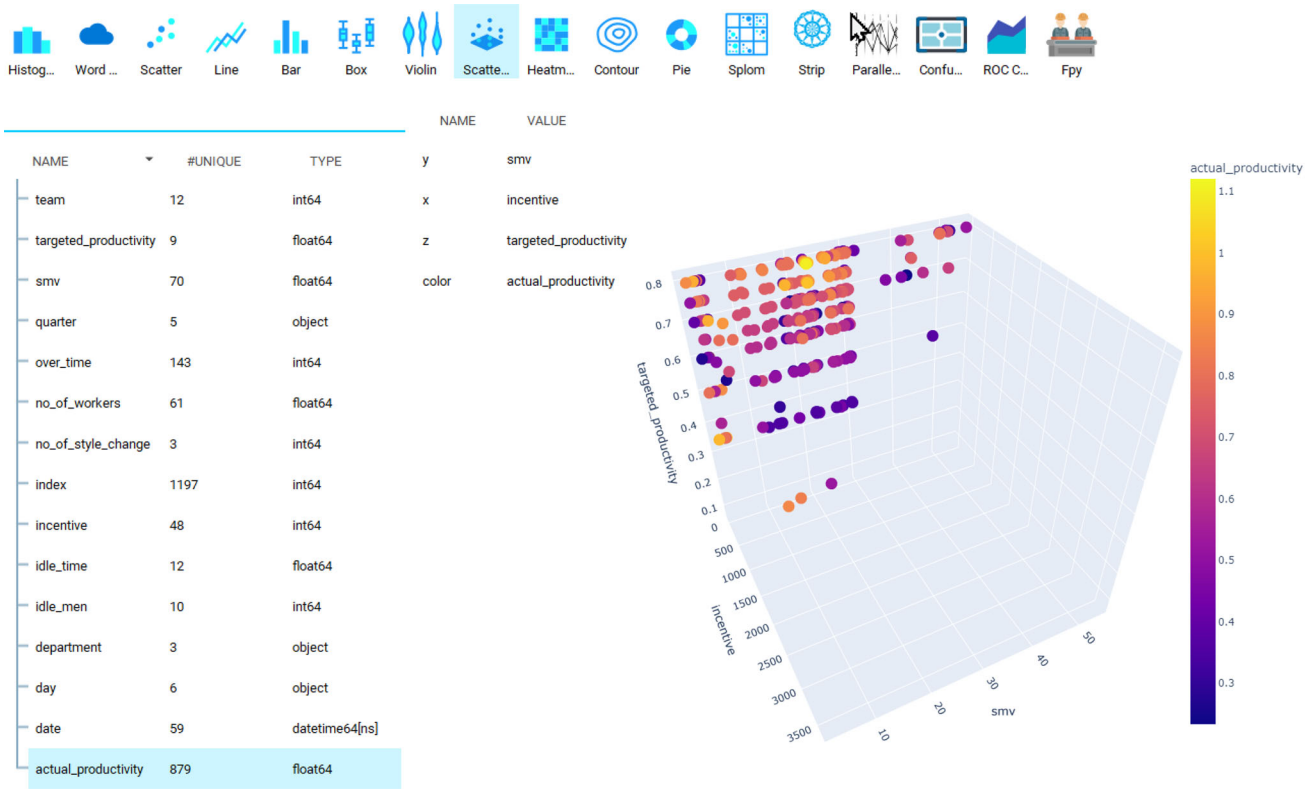

Fig. 1 Graphical user interface by Pandas as a software basis for ML Pro

ML Pro: A digital assistance system for the application of interactive machine learning in production

ML Pro is a DAS for the application of ML to feature tables of production use cases. The DAS was developed in a research cooperation between Bosch Rexroth AG in Homburg and the Chair of Production Systems at the Ruhr-University Bochum and is published as part of this publication. The DAS offers a guidance and an interactive framework for employees of development and planning departments of production companies to apply methods of supervised learning and to make forecasts for declared variables within the production. The software follows the recommendations from the previous chapters for developing a DAS. First, the software basis of the DAS is explained (chapter 3.1). Then, the structure and functionality of the DAS are described (chapter 3.2). In the last step, the functionality and the benefits are illustrated by means of an industrial use case (chapter 3.3).

Software basis of the digital assistance system ML Pro

The Intel Core i7-9850H CPU with 2.6 GHz and 64 GB RAM was used for the calculation of the model trainings and the application of the DAS. Weaker processors can lead to longer waiting times for the application of the methods. The assistance system ML Pro is developed in the Python programming language version 3.7.0. The list of various open-source packages of the virtual environment that gives the DAS its functionality in interaction is presented in Appendix A. In this section, the main packages are discussed. ML Pro is based on the GUI from the Pandas library (PandasGUI). The example in Fig. 1 illustrates the functional capabilities of the PandasGUI. Through the GUI, the basic functionality is given to process tabular data in Pandas-dataframe as matrix format. Accordingly, all mathematical operators and functions for processing the data set of Pandas (version 1.1.4) and NumPy (version 1.20.3) are available. Various analysis graphs can be generated via drag & drop operation in the PandasGUI.

The column names of the current dataframe are shown in a list. In the example shown, a visual correlation analysis is performed with three features. The color scale distinguishes the two classes (0; 1) of the target variables to be classified. Here, the linear influences of the individual features on the target variables are visualized for the user. The example shows that the basic task of data exploration can be accomplished in a user-friendly way by this application of PandasGUI. Using ML algorithms, the multidimensional influences on the target variable can be learned and predictions in regression and classification models can be computed. By linking the Python

Fig. 2 Project management of ML Pro

APIs scikit-Learn, Pycaret, SDV, XGB, LightGBM, CatBoost, Tensorflow, Pytorch, DANet, etc., an extensive variety of DM, DS, ML, and deep learning methods of supervised learning are linked to the PandasGUI. There is a significant synergistic effect of the various Python packages in DAS ML Pro (in Sect. "Model executor, model manager and wrapper manager"). For data visualization, the standard graphs of the library matplotlib [80] and the interactive graphs of the library plotly are used.

Structure of ML Pro

The structure of the DAS is based on the design features of Lee et al. which were introduced in chapter 2.2.3: Project management and user management, resource management, data processing as well as model executor, model manager and wrapper manager (Lee et al., 2019).

Project management and user management

Lee et al. define various requirements for the software structure for user support in ML projects (chapter 2.2.3). Two software modules are project management and user management. The project management screenshot of ML Pro can be seen in Fig. 2. For the employees of the development and planning departments of the manufacturing industry, the application of machine learning methods should not cause any additional organizational effort. Accordingly, the software application starts with the main window of the project structure, which directly accesses the computer-internal folder structure, as shown in Fig. 2. This is where the progress and work results of the ML development processes are recorded. Higher-level, the Project window is activated when the DAS is started and remains permanently open. Project management allows users to create, manage, and control projects with different use cases. Centrally managed functions control the creation, management, and deletion of projects. If a project is created, a *JavaScript Object Notation* (Json) file is generated according to an implemented template, in which meta data of the project as well as project name and the user mode are stored. All information is always stored project specific. This makes it possible to clearly separate use cases and to delete all information about a project when it is no longer needed. Since ML Pro is a software that can be installed locally on a computer, user management is organized by PC access rights. Local users can open only those projects whose data and models are stored in accessible folders. Thus, an additional user management is not required.

At each project design, the DAS prompts to import a tabular data set in one of the different data formats (csv.; xlsx.; db.; pkl.) via simple drag & drop application or alternatively in a folder call. At the start of the project, the target variable and the ML problem for supervised learning are defined in terms of regression and classification. In addition, one of the three user modes "beginner", "advanced" and "expert" is selected depending on the previous experience of the users. The user modes differ on the one hand in the scope of the instructions and on the other hand in the variety of functions. For the beginner mode the AutoML is provided. For the advanced mode and expert mode, the semi-automatic ML pipeline as well as the creation of ANN via buttons are

provided. The expert mode focuses on time-efficient development compared to the advanced mode. The different user modes enable the knowledge transfer from ML experts to ML beginners, since established projects are saved with all development steps (data processing, feature development, training strategy, modeling) as ML models and can be called again in other projects.

Resource management

A resource management regulates the administration of created models and data in a folder system as a further required software component. All developed models, all imported and transformed data, as meta information about data processing steps, the training and all synthetically generated data are managed via a folder system and can be assigned to the respective project via the Json project file.

Data processing

Another requirement defined by Lee et al. is the data processing (Lee et al., 2019). A data processing management is to be implemented to be able to store and trace the data processing steps that are executed on the data set. The data processing steps are stored project-specifically and can be called up independently of the project. The data processing steps are stored in such a way that they can be undone at any time. For this purpose, each function for data processing and for feature engineering is stored centrally in a class and the associated meta data is stored. To be able to undo the data processing steps and the feature engineering functions, they are not immediately applied to the data set when selected by the users. Instead, the meta-data of the data processing function (e.g., deleting a column) or the feature engineering function (e.g., an Anova feature selection) is stored in an instance of a Python class. Each time a new function is added to this class, all the selected functions are executed again on the original imported dataset and this newly configured dataset is made available to the users. This has the benefit that users can deselect the implemented data processing functions and feature engineering functions at any time, and any functions still selected can be re-executed on the original dataset. This gives users a lot of flexibility and transparency as to which functions are applied to the data set.

Model executor, model manager and wrapper manager

In this research, the term ML model is used to refer to the combination of the processing steps of data preparation and feature engineering, and the selection of the algorithm with corresponding HP, as in previous studies of a hydraulic use case by Neunzig et al., (2022b, 2022c). In summary, the model executor, the model manager, and the wrapper manager are described as a defined requirement of the software. The model executor is a functionality that handles the creation of a model, the implementation of the inputs as well as the configurations of the algorithm. The model manager processes additional meta-data for each model on a project-specific basis. A wrapper manager is used to provide functionality from various libraries such as Scikit-Learn or Tensorflow to users without having to program. All models from the Python libraries Sci-kit-Learn, Pycaret, XGBoost, LightGBM, CatBoost, Tensorflow, Pytorch, DANet, etc. are managed in a central Python class. A challenge here is the different API of the packages. The models from different packages must be retrieved in different ways, but the associated data must be stored uniformly and made available to users in a uniform manner. Individual communication functions and storage functions must be written for all packages. The called models must all be storable in a uniform way and models stored as meta data in the Json file must be read in again from the Python libraries when the project is restarted. For both communication directions specific formatting functions were written, which consider the different data types of the packages.

In the following, the structure is described along the execution process in the DAS for the ML model development. The execution process of ML Pro is based on the widely used cross-domain development process CRISP-DM for data mining projects and ML models. The six phases of CRISP-DM are run through in the main windows Data Preparation, Feature Development, Modeling and Model Application. The DAS provides a guided workflow that enables an iterative development process with reversible mathematical operations as well as simultaneous adaptation and visualization of the database. The data table and data visualization can be viewed at any time. The workflow for the ML model development process in DAS is shown in Fig. 3.

By inspecting the dataset after each computed operation, users can assess the effectiveness and correct implementation of each function on a short-cycle basis. It does not run a complete ML pipeline from start to finish. By disabling or repeatedly computing operations in an interactive table, it is possible for users to intermittently make and modify decisions along the ML development process. At any time, the current data set can be viewed in tabular form and visualized in various diagrams. For this purpose, a button is available in the upper right corner of each main window. The progress and the results can be exported to support the documentation of the project.

In the *data preparation* section, essential processing steps of DS and DM are applied to enhance the quality of the dataset through *data cleaning* and *augmentation*. For production, the curse of dimensionality and class imbalance are common challenges that are addressed in data cleansing and data augmentation. In *data cleansing*, various operations are

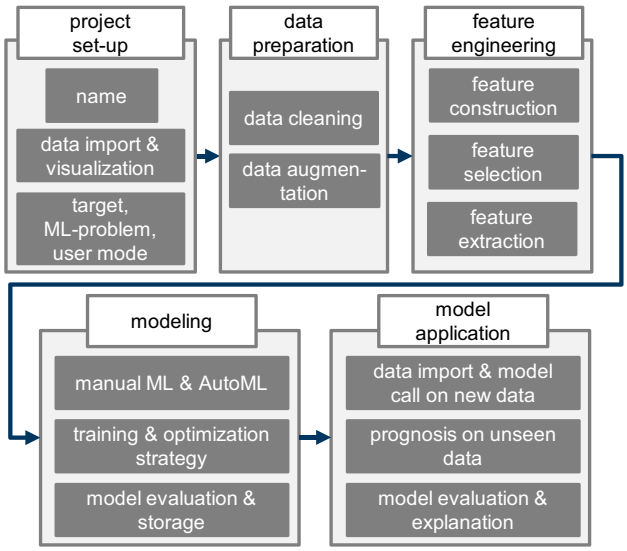

Fig. 3 Workflow for model creation within the assistance system ML Pro

performed to manipulate the columns and rows such as filtering by a criterion with the aim of synchronizing the data types, replacing, or eliminating invalid and implausible cell entries. In addition, users are encouraged to separately check and eliminate "data leakage". If a data set contains relevant data that is not available at the time of prediction of an ML model, so-called "data leakage" or "information leakage" is present. This leads to promising predictions in the training and possibly also in the validation data set, but to insufficient prediction accuracy for unseen data in real production operation due to lack of generalization. In the *data augmentation* section, various oversampling and synthetic data generation procedures can be applied to tabular data sets. This supports especially in compensating for unbalanced distribution and class distribution of a target variable, which are common in the stable processes of production datasets (Neunzig et al., 2022a) In a preliminary study, the suitable methods were compared and established in the DAS: tabular variational autoencoder (Kingma & Welling, 2019), conditional tabular generative adversarial networks (Han et al., 2021), synthetic minority oversampling technique (Chawla et al., 2002). The complicated partitioning of the different subsets of data sets and folds for the CV is performed automatically in the background.

The *feature engineering* section aims to use *feature construction, feature selection*, and *feature extraction* functions to significantly increase the quality of the final dataset for input to the ML algorithm. *Feature construction* is intended to transform or recreate features in a data set. For this purpose, mathematical formulas can be incorporated into the dataset by linking existing features so that additional process and domain knowledge can be considered (Neunzig et al., 2021). Some algorithms require certain transformations of the features prior to inputting the data. For example, scaling numerical features to ranges of values between 0 and 1 in many cases improves prediction accuracy and leads to a reduction in computation time (Nawi et al., 2013). In addition, some algorithms cannot handle categorical features, so they must be transformed into numerical classes. Within feature selection and feature extraction, the aim is to increase predictive power and computational efficiency by eliminating noise using dimensionality reduction.

For the *feature selection* a statistical filter method (Anova), a wrapper method (recursive feature elimination) and an embedded method (XGB) are implemented. By marking the reducing features and entering a target number, the dimension can be applied by clicking buttons without programming. In *feature extraction*, principal component analysis (PCA), independent component analysis (ICA) and linear discriminant analysis (LDA) are implemented. Just like the *feature selection* methods, entering numerical values, marking features, and activating buttons are sufficient to apply the advanced methods to the current data set. For these methods, useful pre-selections for the respective parameters have already been made for the users. The parameters can still be varied to achieve a better result if necessary. In any case, it should be determined to how many features the included data columns should be reduced. The *feature extraction* procedures are implemented in the program in such a way that they can be executed multiple times on the data set and the results of the different procedures can be combined. It is also possible to undo the procedures and run them again later.

In the *modeling* area, the different user modes are distinguished. The model development process is completely automated (AutoML) for the "Beginner" user mode to simplify the use of the program for inexperienced users. Users can select the ML algorithms to be considered for training. Training of the algorithms defaults to fivefold CV to avoid overfitting the models. After the models are trained, users have the option to have the HP of the best model improved with respect to the validation metric using a stand-alone hyperparameter optimization (HPO). For this purpose, the "Tuning" function of Pycaret is called for the HPO.

For the user modes "Advanced" and "Expert", a semi-automatic ML pipeline is implemented in contrast to the user mode "Beginner". The users can bring their own experience into the project with additional functionality and setting options, as shown in Fig. 4. Here, both random search and grid search can be applied. Depending on the algorithm, the matching HPs are loaded directly into the designated window by the wrapper manager. For each HP a certain data type is expected as input. Depending on the data type of the HP, the user can enter either alphanumeric or categorical data. Depending on the grid or random search, it may be necessary

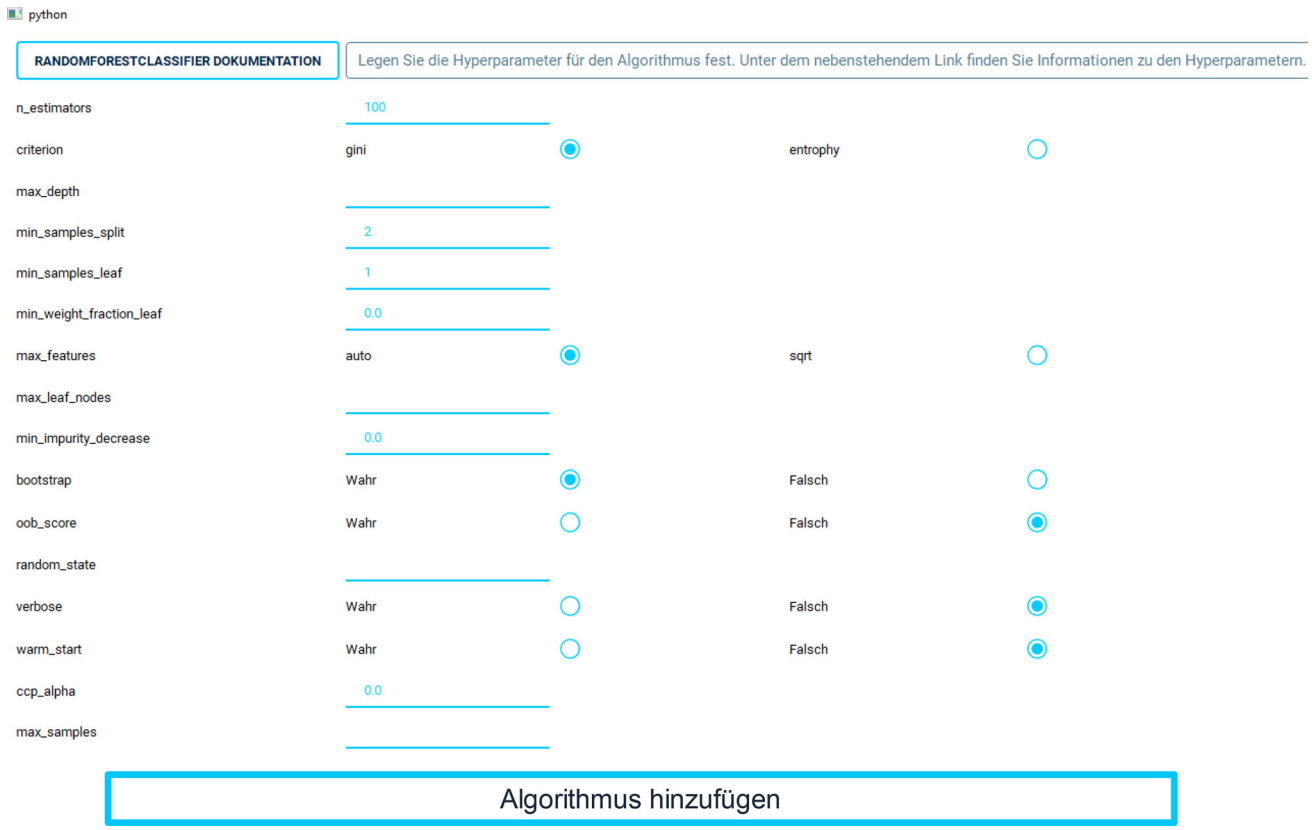

Fig. 4 Hyperparameter optimization using random or grid search

to specify a step size. Activating the "Modell erstellen" button adds the model to the model manager. After calculating the predictive metrics, the algorithms can be recalled individually with HP and their range of values can be adjusted. Users are given the opportunity to make iterative adjustments to the model development process through instant feedback. Experienced users are given the opportunity to provide additional input for the training strategy and the HPO strategy. The training strategy is model-agnostic and centrally managed to ensure comparability of results between models. In each case, the data is split into training and testing data. The ratio for splitting the data sets can be defined. In addition, a CV procedure with different split variants can be applied.

For the development of ANN models, a separate model executor is provided, in which via buttons, layers can be added, adapted, deleted, and visualized in the overall architecture. For the user, an ANN is automatically loaded with sensible default settings. The model builder window for ANN is shown in Fig. 5.

For example, the first layer is automatically added to the model based on the number of features in the dataset. For a binary classification, a sigmoid activation function is preset in the last layer, whereas for a classification with more than two classes, a "softmax" activation function is suitable. For regression problems, a linear activation function is preset in the last layer. The wrapper-manager automatically imports all HP according to their algorithms from the respective libraries. The HP "learning rate" of an ANN expects an input of type "float", i.e. a floating point number. Based on what type of data a model expects for a HP, the user interface for entering the HP is customized for the user. For example, for a floating point number, users are presented with a cell to enter numbers. If a Boolean value is expected, a checkbox is displayed to set the value to True or False. If all models are defined and the HPO strategy and the training strategy are fixed, all models in the model manager in Table 3 are trained when the button "Modelle berechnen" is activated.

The trained models are then displayed in the results table of the model manager in Table 4. The algorithm of the models whose HP are determined by HPO with the best hyperparameter combination, is displayed here. The models can then be evaluated in different ways. Regression or classification specific metrics are displayed for each model. Metrics are displayed for both the training dataset and the unseen test dataset to allow users to detect overfitting. For classification problems, results are presented in a confusion matrix. Results of regressions are visualized in a scatter plot.

In Fig. 6 is the development of ML models and the model evaluation with the explaining text sections in ML Pro displayed. The user language is German.

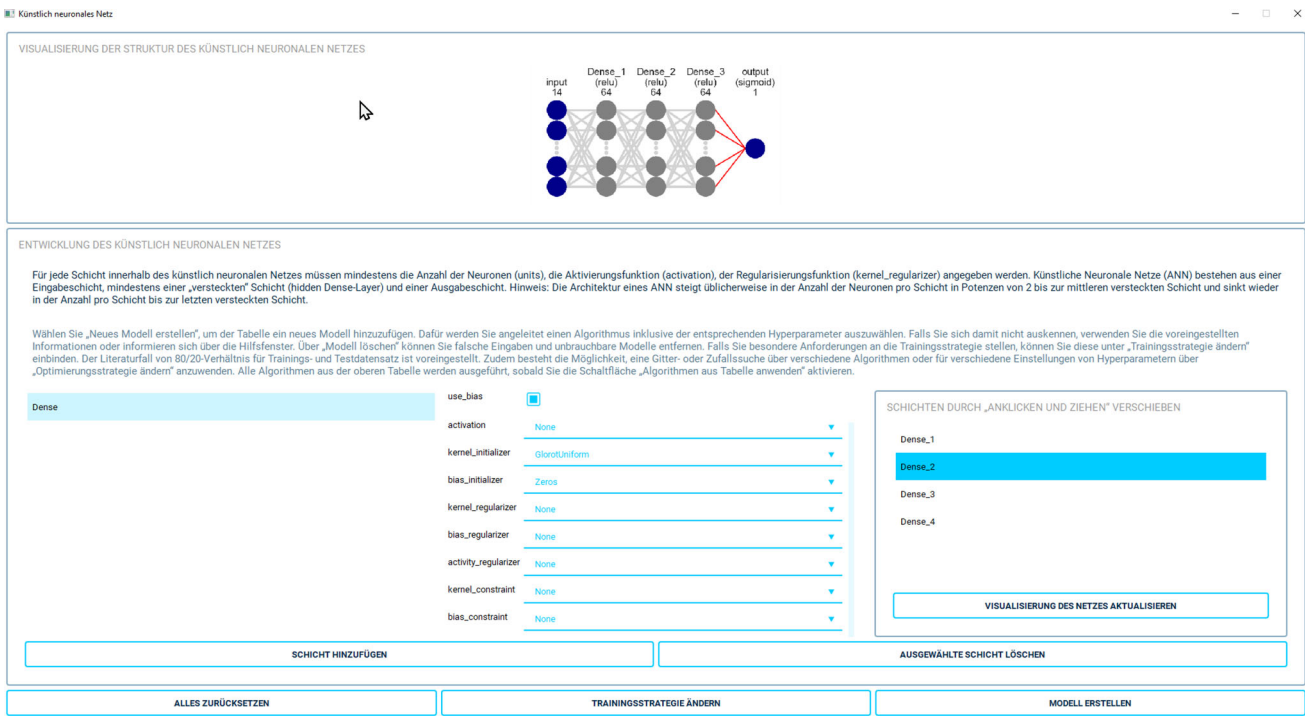

Fig. 5 Model executor for artificial neural networks

Table 3 Model manager: exemplary overview of models

Model name	Algorithm	Date	Hyperparameter
Model_xgb	XGBClassifier	01.09.2022	{Parameter 1: "100"; parameter 2: "False"; etc.}
Model_lgbm	LGBMClassifier	02.09.2022	{Parameter 1: "100"; parameter 2: "False"; etc.}
Model_ANN	ANN_Classifier	02.09.2022	{Parameter 1: "100"; parameter 2: "False"; etc.}

Table 4 Model manager: Exemplary model results for a classification

Model name	Score	Metric	Date	Computation time
Model_xgb	0.81	F1 Score	01.09.2022	1000 s
Model_lgbm	0.83	F1 Score	02.09.2022	1100 s
Model_ANN	0.73	F1 Score	02.09.2022	1800 s

In the *model application* area, predictions are generated on newly added data sets. After training, the models are saved so that they can be recalled in the model application. For this purpose, the new data on which the forecast is to be performed is imported into ML Pro. If it is a regression, the tolerance limits of the specification for the target variable are set by the users, so that an exceeding of the tolerance limits can be indicated. On the newly imported dataset, all processing steps of data preparation and feature development are automatically performed according to the above model definition, which were saved and performed in the project. Accordingly, it is important that both data sets have the same format. If necessary, the processing steps can be deselected and new processing steps can be performed. The data set should then be in the same format with the difference that this one does not have the target variable, but only the characteristics. The predictions of the target variable are presented per identification number in a table on the top left and in a scatter plot on the bottom left. In addition, the results are analytically processed locally for each individual prediction to make the results explainable with explainable ML (EML) approaches. For this explainability ML prediction analysis, the Shapash library is used to determine which feature contributes to the prediction of the target variable and to what extent. In addition, the feature importance of all features can be calculated for each model. Feature importance can only be determined after applying a predictive model. Since the

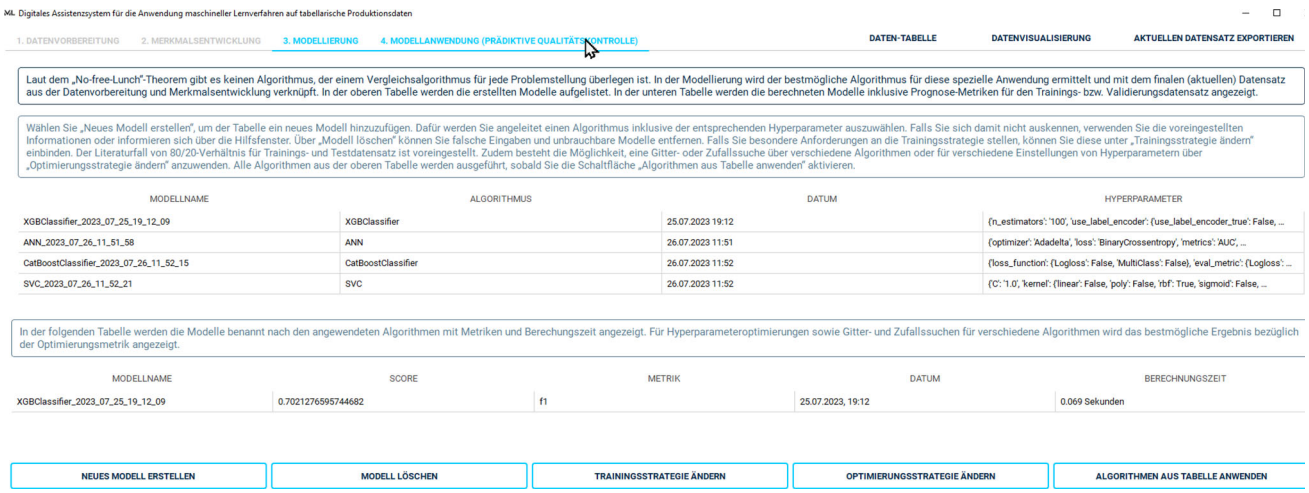

Fig. 6 Model development in ML Pro

calculation of the feature importance requires high calculation times depending on the algorithm, the evaluations of the feature importance for each model must be requested individually by the users. The feature importance is displayed in the program with descending predictive power. Based on the feature importance, users can make further decisions to edit the model or the original database. For example, eliminating features with low predictive power could remove data noise and thus improve the generalizability of the model. Results table also shows which features have the greatest impact on a specific prediction. Activating a data series displays a graph for that data series. This graph shows which feature has the greatest prediction impact locally on the prediction of the target variable from the selected data series.

Applying ML Pro to an industrial use case

ML Pro is applied to an open-source Kaggle production dataset for predicting employee productivity in three iterations to demonstrate the utility and functionality of DAS ML Pro on real data (IshaDS, 2021). The industrial use case has already been studied by Al Imran et al. as a regression problem (Al Imran et al., 2019). Table 5 lists the 14 features and the productivity to be predicted. With respect to the statistical analysis, the following conclusions can be drawn: First, the feature "WIP" contains only 691 values, the rest of the rows are empty. Here, a decision is necessary how to deal with the missing entries. In this case, the function for replacing the invalid columns with the median value of the column provided in the data cleaning section is used. After applying the function, this column contains 1197 values. Second, the maximum of the actual productivity column is greater than one, although the study states that it should be between zero and one. Users have the option to visualize the problem graphically. From the visualization in a histogram, as shown in the data visualization of ML Pro, a small portion of the value range is above productivity level 1. Since the target variable does not necessarily have to be between zero and one, the target variable is not scaled. The characteristics "Quarter", "Department", "Day" and "Date" are not numerical but categorical characteristics. These are converted to numeric features using the ordinal encoder function provided in the feature construction section. In the first iteration, the dimension is not reduced, and no synthetic data is generated.

Table 5 Variables of the industrial use case in ML Pro

Abbreviations	Descriptive texts of the features
wip	"Work in progress", number of unfinished products
team	Categorical numerical value for the number of the team
targeted_productivity	Preset productivity target through leadership
smv	"Standard Minute Value", specified time for a task in minutes
quarter	Quarter of a year
Over_time	Overtime of each team in minutes
No_of_workers	Number of workers
No_of_style_change	Frequency of one work style change per day
Incentive	Level of financial incentive to motivate to work
Idle_time	Downtime
Idle_men	Number of people affected by downtime
Department	Categorical term for the department
Day	Day of the week
Date: Datum	Date of the work day
Actual_productivity	Target variable, actual productivity

Table 6 Statistical analysis in ML Pro

Feature	Type	Amount	N unique	Mean	SD	Min	Max
Wip	float64	691	548	1.190	1837,46	7	23,122
Team	int64	1197	12	6,42	3,47	1	12
Targeted_productivity	float64	1197	9	0,72	0,09	0,07	0,8
SMV	float64	1197	70	15,06	10,94	2,9	54,56
Quarter	object	1197	5	NaN	NaN	NaN	NaN
Over_time	int64	1197	143	4567,46	3348,82	0	25,920
No_of_workers	float64	1197	61	34,61	22,2	2	89
no_of_style_change	int64	1197	3	0,15	0,43	0	2
Incentive	int64	1197	48	38,2	160,18	0	3600
Idle_time	float64	1197	12	0,73	12,71	0	300
Idle_men	int64	1197	10	0,36	3,27	0	45
Department	object	1197	3	NaN	NaN	NaN	NaN
Day	object	1197	6	NaN	NaN	NaN	NaN

Table 7 Overview of the prediction results for the unseen data in ML Pro

Iteration	1	2	3
	R^2-Score	R^2-Score	R^2-Score
LR	− 41.33	–	–
LSR	− 7.38	–	–
RR	− 44.11	–	–
SVR	− 0.109	–	–
XGBoost	0.23	0.30	–
LGBM	0.27	0.31	0.33
RFR	0.32	0.42	**0.454**

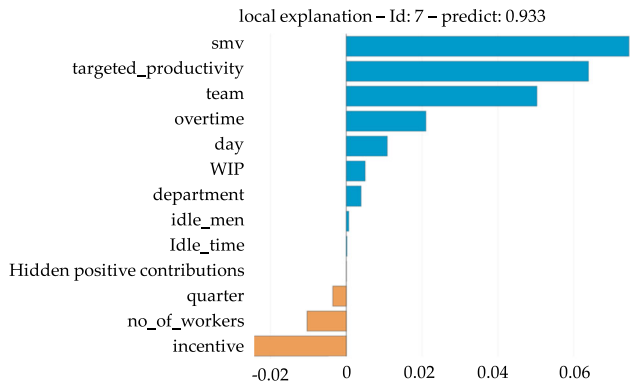

Fig. 7 Local feature importance of the data points of row 7 for the industrial use case in ML Pro

The data were collected in the period from January 2015 to March 2015. Table 6 shows the statistics as they are presented in the program in the main window data preparation.

In the "Beginner" user mode, some algorithms are initialized with default settings. For the model development process in ML Pro, the following range of algorithms with low computation time in the first iteration has been considered: Linear Regression (LR), Lasso Regression (LSR), Ridge Regression (RR), Support Vector Regression (SVR), Random Forest Regressor (RFR), Extreme gradient Boosting (XGB) Regression, and Light gradient boosting Machine (LGBM) Regression. There is no other temporal dependence within the data that needs to be considered for the training strategy. In this respect, a fivefold cross validation with standard training to test data set split is set at 80/20. Table 7 presents the coefficient of determination (R2 score) for assessing accuracy on unseen data for all three iterations of the model development process. From the first iteration, it is clear from negative coefficients of determination that these algorithms are unsuitable for this use case. The estimator performs worse than the estimator to the mean. In the second iteration, synthetic data are included in the prediction using tabular variational autoencoders. All three tree-based procedures improve with synthetic data generation. In an automatic HPO of the tree-based procedures, the coefficient of determination can be increased to 0.454. At this point, the model could be improved with respect to the bias-variance trade-off, but this is not done here.

In the *model application*, 198 rows were taken from the original data set that were previously unseen. These are further considered as unseen "new" data on which the trained RFR model is run. After the data import, all previously described processing steps are automatically executed on the new data set. At this point, the global and local feature weights can be calculated. As an example, a local feature weighting is shown to demonstrate another possibility of investigation. Figure 7 shows the feature weighting of row 7 of the new data set.

It is evident that the top characteristic "smv" influenced the forecast value of productivity for the target value in row 7 most likely to turn out high. In contrast, the feature "incentive" has influenced the prognosis value to turn out low for row 7. The main goal of this application of ML Pro is to show the functionality and usefulness of the program on real production data.

Discussion

This research work aims to present the implementation of a human-centered DAS for the application of IML to universal use on production tabular datasets for employees from development and planning departments of industrial companies. This work has several unique contributions and implications, which are listed below:

(1) It is generically focused on all tabular production data, i.e., other data formats must first be converted to the 2D feature table format. Unlike many other tools, ML Pro is not exclusively tailored to one use case, but is designed to be generic and extensible. Additional Python packages can be integrated into the existing structure. The DAS can also be extended with additional main windows and subcategories to reduce the dimensionality to the feature table in ML Pro.
(2) Both the application of conventional flat ML algorithms like decision trees, gradient trees (XGB, LightGBM, Catboost), support vector machines and the development of self-developed deep learning algorithms (ANN) are enabled in a GUI. New algorithms as well as extensions of existing algorithms can be integrated into the DAS. Furthermore, the processing of image data is already prepared, which can be implemented by integrating additional layer types from Keras into the existing structure. By handling the model manager, multiple models with the same training and optimization strategy can be applied to the data and compared using evaluation metrics.
(3) Different levels of experience and previous knowledge of the user group are considered by implementing three different user modes. The recall of models enables the transfer of knowledge from ML experts to ML advanced and novice users. Furthermore, documentation and traceability are supported.
(4) The complete ML solution development process from importing data from database or other tabular data formats to applying ML models, visualizing, and analyzing predictions is covered by DAS ML Pro. Thus, ML Pro matches the functionality of existing approaches and structures them in an alternative way. Furthermore, models of existing solutions are extended by ML explanatory approaches. Current and advanced methods are provided for the various data preparation and feature engineering steps, and their computation can be reversibly activated and deactivated. After each operation, the data set can be visualized and exported. A demo version for the implementation of the model in real operation is also provided.
(5) By activating buttons and entering parameters the autonomous processes of the DAS are started. In this respect, ML Pro distinguishes itself from graphical approaches that bypass the programming of the ML pipeline by linking function blocks to mathematical operators (GraphicalAI).
(6) In contrast to existing solutions, ML Pro additionally offers the development and integration of methods of synthetic data generation and oversampling procedures for a feature table. The complicated compliance and partitioning of data subsets for the training, validation and testing dataset is calculated in the background of the tool for each dataset.
(7) Explainable machine learning models support confidence in the solution and improve understanding of how the prediction came about (Xie et al., 2023). Therefore, the Shapash module has been incorporated as an explanatory model in ML Pro, as it is not available in any of the available tools.

As any software, ML Pro has certain limitations, which are listed below:

(1) DAS ML Pro is limited to the different tabular formats. The data formats with dimension $d > 2$ must be reduced by projection onto the feature table. For this purpose, it is possible to extend the DAS for further research. This has the advantages that, for example, image and time series data can also be handled with the assistance system if the features have been examined and extracted in the previous step. In this respect, the DAS still has considerable potential for further development by adding such processing steps to the existing system. ML Pro is intentionally designed generically to handle application-independent problems in manufacturing with supervised methods. In the applications we examined, the working memory of 64 GB was by far sufficient. If data sets larger than 1 million lines are processed, the computer system should be designed accordingly strong. The accuracies of the prediction results are identical to those of console computation and have been validated accordingly. The calculation time of the predictions in ML Pro was comparable on average and was, depending on the method considered, about 10 percent in advantage or disadvantage compared to a computation in a Jupyter Notebook.

(2) Supervised learning is the most common-used application in production (Krauss, 2022). Those algorithms of supervised learning are implemented which are listed in Table 9 of Appendix B. Nevertheless, unsupervised learning is very relevant. Oversampling and synthetic data generation methods as well as the feature extraction methods, that are implemented in ML Pro, are mostly based on unsupervised learning. ML Pro should also be extended for the unsupervised and reinforcement learning methods.
(3) Transfer Learning forms an increasingly important role in the domain of production (Wang et al., 2023; Zhao et al., 2022). Series and parallel connection of models is only possible in ML Pro by clever exporting and importing of datasets, and thus not intended in the sense of use. The nesting and majority decision turns out to be very helpful in some use cases. In this respect, an extension of ML Pro is recommended in the future, but the incorporation was not part of the focus for enabling non-expert to apply machine learning in production as a first step.
(4) Approaches of human-centered tools are provided for a user-individual adaptation of the DAS in terms of context sensitivity. This has not been the subject of current research and is recommended for further research. For example, in the case of incorrect use, the DAS could identify the underlying issue and suggest the respective user accordingly. This meta-level of software should be addressed as a separate focus. Both user input and meta-information of the input like speed, degree of logic as well as sensory features like eye movement, or facial expressions could be analyzed.
(5) In the context of this DAS, the ML development process was focused. For ML non-experts, the question is which use cases should be addressed first. There is an opportunity to advise and assist industry users in selecting and assessing use cases. Here, monetary assessment methods of ML solutions have a special value for industry, but there is still a need for research in this area. Furthermore, there is the possibility to provide industry users with further assistance in monitoring ML models. Accordingly, ML-Pro can be extended before and after the ML development process.

In summary, ML Pro contributes to the application of IML for typical production data sets with the following characteristics: high-dimensional, non-stationary, decentralized storage, different data formats, unbalanced data distribution. The DAS is intended for the transfer of knowledge from ML experts to ML novices from development and planning departments of industrial companies, whose backlog of coding skills in the application of IML is overcome by a step-by-step guidance in the procedure model of ML Pro. ML Pro matches the functionality of existing solutions and offers added value through additional functionalities. Furthermore, an alternative approach to GraphicalAI for ML development without coding is presented.

Author contributions Conceptualization, CN, BKand MM; methodology, CN; software, CN; validation, CN, DM; investigation, CN; resources, MM; data curation, CN; writing—original draft preparation, CN; writing—review and editing, CN, DM, BKand MM; visualization, CN; supervision, BK, MM; project administration, MM; funding acquisition, none. All authors have read and agreed to the published version of the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. This research received no external funding. The assistance system ML Pro is not developed for sale nor will be published.

Data availability https://www.kaggle.com/datasets/ishadss/productivity-prediction-of-garment-employees.

Declarations

Competing interests The authors declare no conflict of interest.

Ethical approval Not applicable.

Informed consent Not applicable.

Appendix

This section includes additional information to the described work in two appendices.

Appendix A

Appendix A contains a list of packages (Table 8) including versioning to create a virtual environment that allows the computer to run the *ML Pro* assistance system in a Python development environment (here: Pycharm and Anaconda).

Table 8 Alphabetic list of virtual environment packages for Windows platform win-64

absl-py = 1.2.0
alembic = 1.8.1
altgraph = 0.17.2
appdirs = 1.4.4
argon2-cffi = 21.3.0
argon2-cffi-bindings = 21.2.0
astunparse = 1.6.3
async-generator = 1.10
attrs = 21.4.0
backcall = 0.2.0
beautifulsoup4 = 4.11.1
bleach = 5.0.1
blis = 0.7.8
boruta = 0.3
bosch-ca = 1.0 = 1
brotli = 1.0.9
build = 0.8.0
cachetools = 5.2.0
catalogue = 1.0.0
catboost = 1.0.6
category-encoders = 2.5.0
certifi = 2022.5.18.1
cffi = 1.15.1
charset-normalizer = 2.1.0
chromedriver-py = 105.0.5195.19
click = 8.1.3
cloudpickle = 2.1.0
colorama = 0.4.5
colorlover = 0.3.0
copulas = 0.5.1
cryptography = 37.0.4
ctgan = 0.4.3
cufflinks = 0.17.3
cx-oracle = 8.3.0
cycler = 0.11.0
cymem = 2.0.6
cython = 0.29.14
dash = 2.6.1
dash-bootstrap-components = 1.2.1
dash-core-components = 2.0.0
dash-daq = 0.5.0
dash-html-components = 2.0.0
dash-renderer = 1.8.3
dash-table = 5.0.0
jupyterlab-widgets = 1.1.1
keras = 2.9.0
keras-preprocessing = 1.1.2
kiwisolver = 1.4.4
kmodes = 0.12.1
libclang = 14.0.1
lightgbm = 3.3.2
llvmlite = 0.37.0
mako = 1.2.1
markdown = 3.4.1
markupsafe = 2.1.1
matplotlib = 3.5.2
matplotlib-inline = 0.1.3
missingno = 0.5.1
mistune = 0.8.4
mlflow = 1.27.0
mlxtend = 0.19.0
multimethod = 1.8
murmurhash = 1.0.7
nbclient = 0.6.6
nbconvert = 6.5.0
nbformat = 5.4.0
nest-asyncio = 1.5.5
networkx = 2.6.3
nltk = 3.7
nnv = 0.0.5
notebook = 6.4.12
numba = 0.54.1
numexpr = 2.8.3
numpy = 1.20.3
oauthlib = 3.2.0
opt-einsum = 3.3.0
outcome = 1.2.0
packaging = 21.3
pandas = 1.1.4
pandas-profiling = 3.2.0
pandocfilters = 1.5.0
parso = 0.8.3
patsy = 0.5.2
pefile = 2022.5.30
pep517 = 0.13.0
phik = 0.12.2
pickleshare = 0.7.5
pillow = 9.2.0
pyyaml = 5.4.1
pyzmq = 23.2.0
qhoptim = 1.1.0
qt-material = 2.12
qtpy = 2.1.0
qtwidgets = 0.18
querystring-parser = 1.2.4
rdt = 0.5.3
regex = 2022.7.9
requests = 2.28.1
requests-oauthlib = 1.3.1
rsa = 4.9
ruptures = 1.1.7
scikit-image = 0.19.3
scikit-learn = 0.23.2
scikit-plot = 0.3.7
scipy = 1.5.4
sdmetrics = 0.3.2
sdv = 0.12.0
seaborn = 0.11.2
selenium = 4.4.0
send2trash = 1.8.0
setuptools = 63.4.1
shap = 0.41.0
shiboken2 = 5.15.2.1
six = 1.16.0
sktime = 0.5.3
slicer = 0.0.7
smart-open = 6.0.0
smmap = 5.0.0
sniffio = 1.2.0
sortedcontainers = 2.4.0
soupsieve = 2.3.2.post1
spacy = 2.3.7
sqlalchemy = 1.4.39
sqlparse = 0.4.2
srsly = 1.0.5
statsmodels = 0.13.2
tabulate = 0.8.10
tangled-up-in-unicode = 0.2.0
tenacity = 8.0.1
tensorboard = 2.9.1
tensorboard-data-server = 0.6.1
tensorboard-plugin-wit = 1.8.1

Table 8 (continued)

databricks-cli = 0.17.0
dataclasses = 0.6
debugpy = 1.6.2
decorator = 5.1.1
deepecho = 0.2.0
defusedxml = 0.7.1
docker = 5.0.3
docopt = 0.6.2
entrypoints = 0.4
faker = 4.14.2
fastjsonschema = 2.16.1
flask = 2.1.3
flask-compress = 1.12
flatbuffers = 1.12
fonttools = 4.34.4
funcy = 1.17
future = 0.18.2
gast = 0.4.0
gensim = 3.8.3
ghost-py = 2.0.0.dev0
gitdb = 4.0.9
gitpython = 3.1.27
google-auth = 2.9.1
google-auth-oauthlib = 0.4.6
google-pasta = 0.2.0
greenlet = 1.1.2
grpcio = 1.47.0
h11 = 0.13.0
h5py = 3.7.0
htmlmin = 0.1.12
idna = 3.3
imagehash = 4.2.1
imageio = 2.21.1
imbalanced-learn = 0.7.0
importlib-metadata = 4.12.0
importlib-resources = 5.8.0
install = 1.3.5
ipykernel = 6.15.1
ipython = 7.34.0
ipython-genutils = 0.2.0
ipywidgets = 7.7.1
itsdangerous = 2.1.2
jedi = 0.18.1
pip = 22.1.2
pip-tools = 6.8.0
pipreqs = 0.4.11
plac = 1.1.3
plotly = 5.9.0
pomegranate = 0.14.1
preshed = 3.0.6
prometheus-client = 0.14.1
prometheus-flask-exporter = 0.20.2
prompt-toolkit = 3.0.30
protobuf = 3.19.4
psutil = 5.9.1
pyasn1 = 0.4.8
pyasn1-modules = 0.2.8
pycaret = 2.3.10
pycparser = 2.21
pydantic = 1.9.1
pyee = 8.2.2
pygments = 2.12.0
pyinstaller = 5.2
pyinstaller-hooks-contrib = 2022.8
pyjwt = 2.4.0
pyldavis = 3.2.2
pynndescent = 0.5.7
pynput = 1.7.6
pyod = 1.0.3
pyopenssl = 22.0.0
pyparsing = 3.0.9
pyppeteer = 1.0.2
pyqt5 = 5.15.7
pyqt5-qt5 = 5.15.2
pyqt5-sip = 12.11.0
pyqt5-stubs = 5.15.6.0
pyqtwebengine = 5.15.6
pyqtwebengine-qt5 = 5.15.2
pyrsistent = 0.18.1
pyside2 = 5.15.2.1
pysocks = 1.7.1
python = 3.7.0
python-dateutil = 2.8.2
python-dotenv = 0.20.0
python-graphviz = 0.20
python-qnotifications = 2.0.6
tensorflow = 2.9.1
tensorflow-estimator = 2.9.0
tensorflow-io-gcs-filesystem = 0.26.0
termcolor = 1.1.0
terminado = 0.15.0
text-unidecode = 1.3
textblob = 0.17.1
thinc = 7.4.5
threadpoolctl = 3.1.0
tifffile = 2021.11.2
tinycss2 = 1.1.1
tinydb = 4.7.0
tomli = 2.0.1
torch = 1.12.1
torchaudio = 0.12.1
torchvision = 0.13.1
tornado = 6.2
tqdm = 4.64.0
traitlets = 5.3.0
trio = 0.21.0
trio-websocket = 0.9.2
typing-extensions = 4.3.0
umap-learn = 0.5.3
urllib3 = 1.26.10
vc = 14.2
visions = 0.7.4
vs2015_runtime = 14.27.29016
waitress = 2.1.2
wasabi = 0.9.1
wcwidth = 0.2.5
webdriver-manager = 3.8.3
webencodings = 0.5.1
websocket-client = 1.3.3
websockets = 10.3
werkzeug = 2.1.2
wheel = 0.37.1
widgetsnbextension = 3.6.1
wincertstore = 0.2
wordcloud = 1.8.2.2
wrapt = 1.14.1
wsproto = 1.1.0
xgboost = 1.6.1
xvfbwrapper = 0.2.8

Table 8 (continued)

jinja2 = 3.1.2	pyts = 0.12.0	yacs = 0.1.8
joblib = 1.1.0	pytz = 2022.1	yarg = 0.1.9
jsonschema = 4.7.2	pywavelets = 1.3.0	yellowbrick = 1.2.1
jupyter-client = 7.3.4	pywin32 = 227	zipp = 3.8.1
jupyter-core = 4.11.1	pywin32-ctypes = 0.2.0	
jupyterlab-pygments = 0.2.2	pywinpty = 2.0.6	

Appendix B

Appendix B contains a Table 9 of algorithms that are implemented in the ML Pro assistance system regarding the supervised learning task.

Table 9 Overview of the implemented algorithms in ML Pro

Classification	Regression
	Ridge & linear regression
Gaussian naive bayes	Gaussian naive bayes
Stochastic gradient descent (SGD) classifier	Least absolute shrinkage and selection operator
Support vector machines classification (SVC)	Support vector machines regression (SVR)
Decision tree classifier	Decision tree regressor
Random forest classifier	Random forest regressor
Extreme gradient boosting (XGBoost) classification	Extreme gradient boosting (XGBoost) regression
Categorical boosting (Catboost) classification	Categorical boosting (Catboost) regression
Light gradient boosting machine (LGBM) classification	Light gradient boosting machine (LGBM) regression
Multi-layer perceptron (MLP) classification	
Artificial neural network classifier (ANN)	Artificial Neural Network (ANN) regressor
DANet classification	DANet regression

References

Abadi, M., Agarwal, A., & Barham, P. (2016). *TensorFlow: Large-scale machine learning on heterogeneous distributed systems*. arXiv: 1603.04467.

Al Imran, A., Nur Amin, M., Rifatul Islam Rifat, M., & Mehreen, S. (2019). Deep neural network approach for predicting the productivity of garment employees. In *IEEE 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT'19), Piscataway, NJ*.

Amashaa, M. A., Khairya, D., Abougalalaa, R. A., Alkhalaf, S., & Areed, M. F. (2020). Python-based graphical user interface for automatic selection of data clustering algorithm. *International Journal of Future Generation Communication and Networking, 13*, 451–461.

Amershi, S., Cakmak, M., Knox, W., & Kulesza, T. (2015). Power to the people: The role of humans in interactive machine learning. *Aimag, 35*(4), 105–120. https://doi.org/10.1609/aimag.v35i4.2513

Apt, W., Bovenschulte, M., Priesack, K., Weiß, C., & Hartmann, E. (2018a). Einsatz von digitalen Assistenzsystemen im Betrieb.: Im Auftrag des Bundesministeriums für Arbeit und Soziales. *Forschungsbericht, 502*, 1–125.

Apt, W., Schubert, M., & Wischmann, S. (2018b). Digitale Assistenzsysteme. Perspektiven und Herausforderungen für den Einsatz in Industrie und Dienstleistungen.: Im Auftrag des Bundesministeriums für Arbeit und Soziales. ISBN-13: 978-3-89750-181-2.

Avramidis, E. (2017). QE:GUI—A Graphical User Interface for Quality Estimation. *The Prague Bulletin of Mathematical Linguistics, 109*(1), 51–60. https://doi.org/10.1515/pralin-2017-0038

Bahiuddin, I., Usak, S. A. M., Shapiai, M. I., & Mazlan, S. A. (2017, November). An application of extreme learning machine in a graphical user interface for magnetorheological fluid study. In *2017 International Conference on Robotics, Automation and Sciences (ICORAS)* (pp. 1–5). IEEE. https://doi.org/10.1109/ICORAS.2017.8308064

Bartschat, A., Reischl, M., & Mikut, R. (2019). Data mining tools. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery., 9*(4), 1309. https://doi.org/10.1002/widm.1309

Bashir, A., Awawdeh, M., Faisal, T., & Flower Queen, M. P. (2022). Matlab-based graphical user interface for IOT sensor measurements subject to outlier. In *Advances in Science and Engineering Technology International Conferences (*pp. 1–6). https://doi.org/10.1109/ASET53988.2022.9735063

Bernardo, F. (2020). Interactive Machine Learning for User-Innovation Toolkits—An Action Design Research approach. Goldsmiths Research Online

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., Ohl, P., Thiel, K., & Wiswedel, B. (2009). KNIME-the Konstanz information miner: Version 2.0 and beyond. *AcM SIGKDD Explorations Newsletter, 11*(1), 26–31.

Blutner, D., Cramer, S., Krause, S., Mönks, T., Nagel, L., Reinholz, A., & Witthaut, M. (2007). *Ergebnisbericht der Arbeitsgruppe 5 Assistenzsysteme für die Entscheidungsunterstützung*. SFB 559

Busse, A., Merhar, L., Vernim, S., Kaiser, J., Müller, M., Keller, T., & Korder, S. (2020). Digitale Helfer im Arbeitsalltag.: Praxisleitfaden für Assistenzsysteme in der Produktion. Retrieved August 20, 2022, from https://digitalzentrum-augsburg.de/wp-content/uploads/2020/05/Leitfaden_Digitale_Assistenzsysteme_WEB.pdf.

Carney M, Webster B, Alvarado I, Phillips K, Howell N, Griffith J, Jongejan J, Pitaru A, Chen A (2020) Teachable Machine: Approachable Web-Based Tool for Exploring Machine Learning Classification. CHI 2020, Honolulu, HI, USA:1–8. doi:https://doi.org/10.1145/3334480.3382839

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Csiszar, A. Hein, P., Wachter, M., Verl, A., & Bullinger, A. C. (2020). Towards a user-centered development process of machine learning applications for manufacturing domain experts. In *2020 Third International Conference on Artificial Intelligence for Industries (AI4I)* (pp. 36–39). IEEE

Danyluk, A., & Buck, S. (2019). Artificial intelligence competencies for data science undergraduate curricula. *AAAI, 33*, 9746–9747. https://doi.org/10.1609/aaai.v33i01.33019746

Dehnbostel, P., Richter, G., Schröder, T., & Tisch, A. (Eds.). (2021). *Kompetenzentwicklung in der digitalen Arbeitswelt: Zukünftige Anforderungen und berufliche Lernchancen* (1st ed.). Schäffer-Poeschel Verlag, Stuttgart.

Demsar, J., Curk, T., Erjavec, A., Gorup, C., et al. (2013). Orange: Data mining toolbox in Python. *Journal of Machine Learning Research, 14*, 2349–2353.

DIN 9241 Ergonomie der Mensch-System-Interaktion.: Grundsätze der Informationsdarstellung EN ISO 9241-110:2020(EN ISO 9241-110:2020)

Dogan, A., & Birant, D. (2021). Machine learning and data mining in manufacturing. *Expert Systems with Applications, 166*(114060), 1–22. https://doi.org/10.1016/j.eswa.2020.114060

Doty, C., Gallagher, S., Cui, W., Chen, W., Bhushan, S., Oostrom, M., Akers, S., & Spurgeon, S. R. (2022). Design of a graphical user interface for few-shot machine learning classification of electron microscopy data. *Computational Materials Science, 203*(3–4), 1–19. https://doi.org/10.1016/j.commatsci.2021.111121

Dudley, J. J., & Kristensson, P. O. (2018). A review of user interface design for interactive machine learning. *ACM Transactions Interact. Intell. Syst., 8*(2), 1–37. https://doi.org/10.1145/3185517

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. Advances in neural information processing systems (Vol. 28)

Filz, M.-A., Bosse, J. P., & Herrmann, C. (2023). Digitalization platform for data-driven quality management in multi-stage manufacturing systems. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-023-02162-9

Han, G., Liu, S., Chen, K., Yu, N., Feng, Z., & Song, M. (2022). Imbalanced sample generation and evaluation for power system transient stability using ctgan. In *Intelligent Computing & Optimization: Proceedings of the 4th International Conference on Intelligent Computing and Optimization 2021 (ICO2021) 3* (pp. 555–565). Springer International Publishing.

Isha, D. S. (2021). Productivity Prediction of Garment Employees. Retrieved September 22, 2022, from https://www.kaggle.com/datasets/ishadss/productivity-prediction-of-garment-employees.

Jovic, A., Brkic, K., & Bogunovic, N. (2014). An overview of free software tools for general data mining. In *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 1112–1117). IEEE. https://doi.org/10.1109/MIPRO.2014.6859735

Kaji, F., Nguyen-Huu, H., Narayanan, J. A., Zimny, M., & Toyserkani, E. (2023). Intermittent adaptive trajectory planning for geometric defect correction in large-scale robotic laser directed energy deposition based additive manufacturing. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-023-02194-1

Khalajzadeh, H., Simmons, A. J., Abdelrazek, M. (2020). End-user-oriented tool support for modeling data analytics requirements. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. https://doi.org/10.1109/VL/HCC50065.2020

Kingma, D. P., & Welling, M. (2019). An Introduction to Variational Autoencoders. *FNT in Machine Learning, 12*(4), 307–392. https://doi.org/10.1561/2200000056

Klemm, S., Scherzinger, A., Drees, D., Jiang, X. (2018). Barista—A graphical tool for designing and training deep neural networks:1–8

Krauss, J. (2022). *Optimizing hyperparameters for machine learning in production*. Dissertation,. APPRIMUS Wissenschaftsverlag

Krüger, J., Fleischer, J., Franke, J., & Groche, P. (2019). *WGP-Standpunkt: KI in der Produktion: Künstliche Intelligenz erschließen für Unternehmen*. Wissenschaftliche Gesellschaft für Produktionstechnik e.V

Lee, K., Yoo, J., Kim, S.-W., & Hong, J. (2019). Autonomic machine learning platform. *International Journal of Information Management, 49*, 491–501. https://doi.org/10.1016/j.ijinfomgt.2019.07.003

Link, M., & Hamann, K. (2019). Einsatz digitaler Assistenzsysteme in der Produktion. *ZWF, 114*(10), 683–687. https://doi.org/10.3139/104.112161

Mättig, B., & Kretschmer, V. (2019). Einsatz digitaler Assistenzsysteme in der Logistik 4.0. In: Hompel M ten, Vogel-Heuser B, Bauernhansl T (Eds.), Handbuch Industrie 4.0 (pp. 1–25). Springer

Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., & Euler, T. (2006, August). Yale: Rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1–935). ACM Press

Milde, S., Liebgott, A., Ziwei, W., Wenyi, F., Jiahuan, Y., Lukas, M., Petros, M., Fabian Bamberg, Konstantin Nikolaou, Sergios Gatidis, Fritz Schick, Bin Yang, Thomas Kustner (2018) 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC): Proceedings November 12–15, 2018, Honolulu, Hawaii, USA:1–10.

Mishra, M., & Srivastava, M. (2014). A view of Artificial Neural Network. In *IEEE International Conference on Advances in Engineering & Technology Research (ICAETR - 2014)*, (pp. 1–3). https://doi.org/10.1109/ICAETR.2014.7012785

Mishra, P., Roger, J. M., Rutledge, D. N., Biancolillo, A., Marini, F., Nordon, A., & Jouan-Rimbaud-Bouveresse, D. (2020). MBA-GUI: A chemometric graphical user interface for multi-block data visualisation, regression, classification, variable selection and automated pre-processing. *Chemometrics and Intelligent Laboratory Systems, 205*, 1–12. https://doi.org/10.1016/j.chemolab.2020.104139

Moore, J. H. (2018). Information about automated machine learning (AutoML). Retrieved September 23, 2022, from, https://automl.info/.

Na, G.-Y., & Yang, J. (2023). Two-dimensional polygon classification and pairwise clustering for pairing in ship parts nesting. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-023-02196-z

Naik, A., & Samant, L. (2016). Correlation review of classification algorithm using data mining tool: WEKA, Rapidminer, Tanagra, Orange and Knime. *Procedia Computer Science, 85*, 662–668. https://doi.org/10.1016/j.procs.2016.05.251

Nawi, N., Atomi, W., & Rehman, M. (2013). The effect of data pre-processing on optimized training of artificial neural networks. *Procedia Technology, 11*, 32–39. https://doi.org/10.1016/j.protcy.2013.12.159

Neunzig, C., Fahle, S., Kuhlenkötter, B., & Möller, M. (2021). Feature engineering for a cross-process quality prediction of an end-of-line hydraulic leakage test using an experiment sample. In: Herberger, D., Hübner, M. (Eds.), *Proceedings of the Conference on Production Systems and Logistics 2021* (pp. :156–166). Publish-Ing.

Neunzig, C., Fahle, S., Kuhlenkötter, B., Möller, M., Schulz, J(2022a). Approach to data pre-processing for predictive quality of hydraulic test results in a dynamic manufacturing environment: Concept drift approach with ensemble classification for abrupt shift correction. In: VDI Wissensforum GmbH (ed) 23. Leitkongress der Mess- und Automatisierungstechnik: Automation 2022.: Automation creates Sustainability. VDI Verlag:425–437.

Neunzig, C., Fahle, S., Schulz, J., Möller, M., & Kuhlenkötter, B. (2022b). Approach to a decision support method for feature engineering of a classification of hydraulic directional control valve tests (pp. 1–11). https://doi.org/10.15488/12177

Neunzig, C., Fahle, S., Schulz, J., Möller, M., & Kuhlenkötter, B. (2022c). Model selection for predictive quality in hydraulic testing. In *Procedia CIRP(107)* (pp. 320–325). https://doi.org/10.1016/j.procir.2022.04.052

Ng, C., Bruestle, J., & Retford, B. (2022). Vertex AI.: Rapidly build, deploy, and scale ML models with pre-trained and custom tools on a unified artificial intelligence platform. Retrieved October 10, 2022, from https://cloud.google.com/vertex-ai#section-16

Ogunleye, G., Fashoto, S., Daramola, C. Y., Ogundele, L. A., Ojewumi, T. O., & Timilehin, A. (2019). 3770 Published By: Blue Eyes Intelligence Engineering & Sciences Publication Retrieval Number: B3426078219/19©BEIESP. https://doi.org/10.35940/ijrte.B3426.078219 Journal Website: www.ijrte.org Development of a Simple Graphical Interface Based Software for Machine Learning and Data Visualization. IJRTE 9(2):3770–3777. doi:https://doi.org/10.35940/ijrte.B3426.078219

Pokorni, B., Braun, M., & Knecht, C. (2021). Menschzentrierte KI-Anwendungen in der Produktion.: Praxiserfahrungen und Leitfaden zu betrieblichen Einführungsstrategien. https://www.researchgate.net/profile/Martin-Braun-6/publication/,: 24–29.

Rosemeyer, J., Bardy, S., Marta, P., Bosani, E., Schubert T, & Metternich, J. (2022). Towards artificial intelligence in production: A competence profile for shop floor managers. In *12th Conference on Learning Factories CLF2022:1–6*

Santos-Pereira, J., Gruenwald, L., & Bernardino, J. (2022). Top data mining tools for the healthcare industry. *Journal of King Saud University - Computer and Information Sciences, 34*(8), 4968–4982. https://doi.org/10.1016/j.jksuci.2021.06.002

Shen, A., & Sun, Y. (2021). GraphicalAI: A user-centric approach to develop artificial intelligence and machine learning applications using a visual and graphical language. In *2021 4th International Conference on Data Storage and Data Engineering* (pp. 52–58). https://doi.org/10.1145/3456146.3456155

Slater, S., Joksimović, S., Kovanovic, V., Baker, R., & S., Gasevic D,. (2017). Tools for Educational Data Mining: A Review. *Journal of Educational and Behavioral Statistics, 42*(1), 85–106. https://doi.org/10.3102/1076998616666808

Spain, G.,R. (2017) Dzone's guide to artificial intelligence: machine learning & predictive analytics.: Key Research Findings. https: //dzone.com/storage/assets/6907133-dzone-guide-artificialintelligence-2017.pdf

Thornton, C., Hutter, F., Hoos, H., Leyton-Brown, K. (2012). Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms.

VDI Wissensforum GmbH (ed). (2022). 23. Leitkongress der Mess- und Automatisierungstechnik: Automation 2022.: Automation creates Sustainability. VDI-Berichte 2399. VDI Verlag

Wang, P., Wang, T., Yang, S., Cheng, H., Huang, P., & Zhang, Q. (2023). Production quality prediction of cross-specification products using dynamic deep transfer learning network. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-023-02153-w

Witten, I. H., Frank, E., Hall, M. A., Pal, C. J. (2016) The WEKA Workbench.: Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques". Retrieved September 20, 2022, from https://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf.

Xames, M. D., Torsha, F. K., & Sarwar, F. (2023). A systematic literature review on recent trends of machine learning applications in additive manufacturing. *Journal of Intelligent Manufacturing, 34*(6), 2529–2555. https://doi.org/10.1007/s10845-022-01957-6

Xie, S., He, Z., Loh, Y. M., Yang, Y., Liu, K., Liu, C., Cheung, C. F., Yu, N., & Wang, C. (2023). A novel interpretable predictive model based on ensemble learning and differential evolution algorithm for surface roughness prediction in abrasive water jet polishing. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-023-02175-4

Yamamoto, S. (ed) (2017). Human interface and the management of information supporting learning, decision-making and collaboration: 19th International Conference, HCI International 2017, Vancouver, BC, Canada, 2017, Proceedings, Part II, 1st edn. Information Systems and Applications, incl. Internet/Web, and HCI 10274. Springer International Publishing; Imprint: Springer, Cham

Zhang, D., Maslej, N., Brynjolfsson, E., Etchemendy, J., Lyons, T., Manyika, J., Ngo, H., Niebles, J., Sellitto, M., Sakhaee, E., Shoham, Y., Clark, J., & Perrault, R. (2022). *The AI Index 2022 Annual Report* (pp. 143–171). AI Index Steering Committee, Stanford Institute for Human-Centered AI, Stanford University.

Zhao, K., Jiang, H., Wu, Z., & Lu, T. (2022). A novel transfer learning fault diagnosis method based on Manifold Embedded Distribution Alignment with a little labeled data. *Journal of Intelligent Manufacturing, 33*(1), 151–165. https://doi.org/10.1007/s10845-020-01657-z

Zöller, M.-A., Titov, W., Schlegel T, Huber, M. F. (2022) XAutoML: A Visual Analytics Tool for Establishing Trust in Automated Machine Learning. ACM Comput. Surv.:1–37