Expert System for the Machine Learning Pipeline in Manufacturing

M. Frye*, J. Krauß*
R. H. Schmitt*/**

* Fraunhofer Institute for Production Technology IPT, Steinbachstraße 17, Aachen 52074, Germany
** Laboratory for Machine Tools WZL RWTH Aachen University, Cluster Production Engineering 3A 540, Aachen 52074, Germany

Abstract: Many success stories already exist with regard to the implementation of Machine Learning (ML) and Artificial Intelligence (AI) in manufacturing. However, companies with traditional focus on production technologies face challenges in conducting AI-projects successfully and lack knowledge of which obstacles may occur and how to decide in the implementation phase. In this paper, we develop an approach that focuses on the methodological necessary steps for the successful application of ML and AI in manufacturing. Optimization potentials and decisions to be made are outlined in every step. A main focus is put on optimizing hyperparameters of ML-models as one promising approach for improving overall ML-model performance. An expert system is presented that enables the selection of suitable hyperparameter optimization techniques. The concept is validated based on manufacturing of compressor components of a turbofan engine.

Copyright © 2021 The Authors. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0)

Keywords: Machine Learning; Artificial Intelligence; Manufacturing; Production; Expert Systems; Hyperparameter Optimization.

1. INTRODUCTION

The application of Machine Learning (ML) and Artificial Intelligence (AI) has already produced numerous optimizations in many different domains. In particular, manufacturing benefits from the use of ML and AI. Shortly explained, ML learns from historical data and produces a result that enables the development of an AI-system. Best practices of ML-implementations in production environment are well known and are already classified in different application areas (Krauß et al. 2019b). This classification comprises three different areas “Process”, “Product” and “Machines and Assets”. Exemplarily the application area “Process” can be further divided into fields such as “Predicitive Process Control” (PPC). PPC consists of the control and optimization of production processes. An exemplary use-case in this field is the prediction of product quality in order to increase process efficiency and reduce the amount of scrap (Krauß et al. 2019a).

Even though there are already many success stories in manufacturing, the methodological implementation of AI-projects remains very challenging. Especially companies with traditional focus on production technologies face the challenge of conducting AI-projects successfully. Companies lack experience in dealing with obstacles as well as numerous manual decisions that highly depend on experience and expert knowledge. Especially manual decisions within the implementation phase can form the basis for identifying most promising potentials for improving final ML-model performance, which is crucial for the success of an AI-project. Potentials can range from the choice of data preprocessing methods or suitable ML-algorithms. In particular, optimizing hyperparameters of ML-algorithms enables significant leap in ML-model performance, while being challenging especially for companies with production background.

For these reasons, we present a comprehensive concept that outlines the necessary steps for the successful application of ML and AI in manufacturing. Within each step, the decisions to be made and optimization potentials are pointed out. A main focus is put on hyperparameter optimization (HPO) due to the potential of improving ML-model performance considerably. An expert system (ES) is introduced, which supports in the selection of suitable HPO-techniques for a given use-case. The concept is validated based on a milling process of jet engine components, in which process parameters are predicted.

2. LITERATURE REVIEW

First, literature is reviewed regarding existing concepts for implementing ML in manufacturing. Afterwards, an overview about hyperparameter optimization is given.

2.1 Methodological Implementation of AI-Projects

General concepts already exist in terms of the methodological implementation of AI-projects. The most common approach is the Cross-Industry Standard Process for Data Mining (CRISP-DM). Within six iterative phases, the project is conducted (Chapman et al. 1999). Besides CRISP-DM, similar approaches have been introduced to methodologically carry out AI-projects. Examples range from Knowledge Discovery in Databases, also called KDD, to SEMMA, which is short for Sample, Explore, Modify, Modell, Assess. These approaches are most widely used. Based on CRISP-DM, further developments have led to the approach Cognitive Project Management for AI (CPMAI) (Cognilytica).
CPMAI mainly attempts to meet AI-specific requirements, which are e.g. simultaneous ML-algorithm evaluation and hardware-centric model deployment. In a similar way, methodologies such as Team Data Science Process (TDSP) focus on the implementation of ML and AI (Microsoft Azure 2020). However, CPMAI and TDSP are still very generic, do not take into account domain-specific knowledge and do not focus on specific optimization potentials and manual decisions being made during individual implementation phases.

2.2 Hyperparameter Optimization

Hyperparameter optimization (HPO) or hyperparameter tuning is the problem of finding the hyperparameter combination over the search space that, when applied to an algorithm to be trained, returns the model with the best quality, e.g. lowest loss function. This procedure, however, is not simple as manually exploring the search space by trial and error is tedious and tends to lead to unsatisfactory outcomes, since usually solving hyperparameter optimization problems can demand a large amount of time and computing power (Hutter et al. 2011). Given a learning algorithm A, a limited amount of training data \( D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \) and sets of hyperparameters \( \lambda_1, \lambda_2, \ldots, \lambda_m \) with domains \( \Lambda_1, \Lambda_2, \ldots, \Lambda_m \) inside a space \( \mathcal{A} \), the goal is to determine \( \lambda^* \) (1) with optimal generalization performance. Among the approaches to calculate generalization performance, splitting \( D \) into \( k \) equal sized disjoint training and validation sets, \( D_{\text{train}}^{(i)} \) and \( D_{\text{valid}}^{(i)} \), and carrying \( k \) performance evaluations, known as k-fold cross validation (Ron Kohavi 1995). This way, hyperparameter tuning can be written as:

\[
\lambda^* = \arg\min_{\lambda \in \mathcal{A}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A_\lambda, D_{\text{train}}^{(i)}, D_{\text{valid}}^{(i)})
\]  

where \( \mathcal{L}(A_\lambda, D_{\text{train}}^{(i)}, D_{\text{valid}}^{(i)}) \) is the loss (e.g. misclassification rate) achieved by \( A \) when trained on \( D_{\text{train}}^{(i)} \) and evaluated on \( D_{\text{valid}}^{(i)} \) (Thornton et al. 2013). Considering \( \mathcal{L}(\cdot) \) as a “black box function”, meaning it is hard to predetermine its behaviour, a common strategy is to select combinations \( \{\lambda_1, \lambda_2, \ldots, \lambda_m\} \) to be evaluated, from which the best is considered \( \lambda^* \). This selection is considered the critical step in hyperparameter optimization (HPO) (James Bergstra und Yoshua Bengio 2012).

The HPO-problem is in general non-smooth, since a significant number of hyperparameters are discrete (e.g. the choice of 11 vs. 12 penalization for support vector machines). In general, the HPO-problem is also non-convex and can be described as stochastic or noisy, since most underlying algorithms are heuristic approximations with variance in regards to the final output. (Stack Overflow 2017) The key-challenges in solving the HPO-problem are: costly objective function evaluations, randomness and complex search space (Marc Claesen und Bart De Moor 2015). Therefore, applying HPO-techniques instead of selecting hyperparameters manually leads to an increase of model performance (Wang-Chi Cheung et al. 2018).

Given these challenges, different techniques for hyperparameter selection were created and can be classified. Apart from other means of categorizing specific tuning methods, an approach can be to differentiate by the two following characteristics:

- Model-based vs. Non model-based
- Using low-fidelity approximations to evaluate a hyperparameter configuration vs. fully evaluating a hyperparameter configuration

The impact that the selection of the HPO-technique has on the final performance is significant. Performance improvements of over 20% are described in various publications (Wang-Chi Cheung et al. 2018; AbdElRahman ElSaid et al. 2018). In regards how to select an HPO-technique, the literature just provides general guidelines, that do not consider domain expertise and past implementation experiences (Feurer und Hutter 2019).

3. METHODOLOGY

In this chapter, the concept for a structured implementation of ML and AI is pointed out. A focus is made on the development of an ES, which supports in the selection of suitable HPO-techniques.

3.1 Requirements of the Manufacturing

Many different requirements influence the use of ML and AI within manufacturing. The requirements can be classified in use-case, data set, external and model requirements. In regards to data set requirements, many different sensors are used that acquire data in different frequencies and starting times. Production data comes often in uncleaned and imbalanced form. The appropriate order of computing power and time are challenging requirements in manufacturing. Since production experts aim to understand the results of ML-models, transparent and less complex models are required. Moreover, the availability of noisy data requires robust learners.

3.2 Structured Implementation of ML and AI

Existing methodologies such as CRISP-DM are very generic and do not take into account relevant domain-specific knowledge. Especially in production, domain expertise is required, since an expert needs to support data scientists in decision-making during AI-projects. For these reasons, in Figure 1, we present a holistic ML-pipeline that covers all relevant phases of an AI-project by considering data scientist, IT-expert and production expert. The presented ML-pipeline is based on many years of project experience.

**Use-Case Selection.** As a very first step of an AI-project, the use-case is chosen. For selecting suitable use-cases, the profitability is calculated and strategic relevance, stakeholder commitment, existing infrastructure as well as data available are evaluated. Finally, AI-use-cases are prioritized. The output of the first phase are selected AI-use-cases.

**Data Integration.** Based on prioritized AI-use-cases, acquired data needs to be integrated. In practice, integrating
data from different sources coming in many different structures poses a major challenge. In manufacturing, time series data is existent in many cases. If data sources have different time frequencies and shifts, time needs to be synchronized. The decision about the integration strategy requires a strong collaboration between production expert and data scientist, since production experts know about relevant process parameters to be covered and its location. Based on this information, a data scientist is able to integrate the data into unified storage. The outcome is data that is integrated from various sources to one storage.

**Figure 1: ML-Pipeline in Production**

**Data Preparation.** Integrated data from previously selected AI-use-cases can still be noisy, while containing missing or constant values through the occurrence of different machines and other measurements. Poor data quality requires the preparation for further modelling. Based on an initial data quality check, the preprocessing steps that need to be performed are determined. A structured data preprocessing pipeline considers cleaning, transformation, reduction as well as augmentation and balancing. In each preprocessing step, a decision must be made on the method to be used. Data cleaning comprises the decision of which method to be applied for handling missing values, outliers and noisy data such as duplicate rows. Subsequently, data needs to be transformed, if data has different types and different ranges. Afterwards, in case of high dimensionality, data should be reduced through the approaches of dimensionality reduction and feature selection. On the other hand, data can be too small, which requires data augmentation. The vast majority of production use-cases, especially with respect to the prediction of product quality, exhibit an imbalance between the different classes to be predicted that requires balancing. Since the final model performance is dependent on the methods being used during preprocessing, this phase is highly iterative with modelling. A very important and additional step is feature engineering that requires the production expertise in order to identify valuable features for further modelling. Ultimately, data is of high quality through preparation. As a consequence of being such an important but also demanding task, in practice, there is often a separate role for data preparation, the data engineer.

**Modelling.** Based on the prepared data set, the first step of modelling comprises the selection of suitable ML-algorithms. Through the consideration of use-case requirements, previously performant ML-algorithms on similar use-cases, ML-algorithm requirements and domain expertise, ML-algorithms can be selected (Krauß et al. 2019a). The chosen ML-algorithms can then be implemented. During training of ML-algorithms, a crucial step and possibility of significant improvement of the performance is the optimization of existing hyperparameters. After model training, models can be assessed and further improvement potentials can be derived. Modelling results are also used to optimize data preparation steps and introduce further features to the model showing the highly iterative character of the ML-pipeline. The outcome of the modelling phase is a performant ML-model, which is deployable in the production environment. For the successful conduction of data preparation and modelling, the **Data and Process Understanding** is mandatory. In practice, a process and data understanding requires a mutual qualification of production, IT and data scientist expertise. The output is a performant ML-model that can be deployed in the next phase.

**Deployment.** The results of the ML-model can be used for different applications. It is therefore determined, how the ML-model is applied in production. One possible deployment of ML-models is to enable an automated decision. Another use is a support for decision makers. In case of product quality prediction, it can be a support in form of an information that e. g. a product is out of specification with a probability of 90 %. The deployment design comprises a decision whether ML-models are trained online or offline. In addition, ML-models can be implemented in different software architectures. A decision also needs to be made on the strategy, whether one, multiple or competing models are serving outputs. Once the deployment design is finished, the software system is productionized and tested. After initial deployment, the life cycle of the whole software system needs to be managed that requires monitoring of system’s health. A model retraining strategy is determined. The output is a deployed software system with ML-models that are continuously managed.

**Certification.** Starting from a deployed software system with implemented ML-models, this phase consists of the development of a certification strategy. Based on an initial description of process and behaviour of an AI-system, contact relevant certification are contacted. The certification aspect comprises the internal audit, following by a pre- and final audit of the AI-system, ending up in a certified AI-system.

### 3.3 ES for the selection of HPO-Techniques

Data integration, data preparation, modelling and deployment include the selection of specific techniques. In data preparation, data encoding and normalization techniques might be applied while in modelling, the ML-algorithm as well as HPO-technique need to be selected. Recommendations, which technique to select or to exclude enables the reduction of the search space through a “warm start”. We chose to apply the concept of rule-based ES to create these recommendations (Eric J. Horvitz et al. 1988), since it has shown great success in similar problems. Furthermore, rule-based expert systems are updatable, transparent, and can be set-up with limited empiric data: The database is continuously updated in regards...
to the specific use-case. Recommendation logics are stored in the Knowledge Base. An inference engine uses this data to provide recommendations, based on logics that are stored in the ES (see Figure 2). By describing the current use-case, the knowledge database is filled with initial information. Based on this, the inference engine provides a recommendation for the data integration phase. After conducting data integration, results are stored in the knowledge database, which is repeated for the different phases of the ML-pipeline. The concept of the ES can be used by AutoML-systems as well as for manual technique selection, conducted by a data scientist.

Figure 2: Expert System for ML-Pipeline

In the following, we describe the ES, which focuses on the HPO-technique selection. The design is based on domain knowledge and evaluations of historic use-cases. We chose the approach of belief rule based (BRB) ES (Jian-Bo Yang et al. 2006; Xu 2012) since they can handle qualitative as well as quantitative data (Leilei Chang et al. 2013). BRB has been successfully applied to similar problems (Dong-Ling Xu et al. 2007; Swati Sachan et al. 2020). To set the rules in the ES, HPO-techniques need to be benchmarked, which was done by considering:

- **Strong Anytime Performance:** Since available resources often only suffice for the full training and evaluation of a handful of ML-models, the HPO-method should be able to find good hyperparameters within few evaluations.
- **Strong Final Performance:** The performance of the ML-model trained on the best hyperparameter configuration presented by hyperparameter tuning methods is crucial.
- **Effective Use of Parallel Resources:** Since parallel resources are becoming more common to use and easier to obtain, hyperparameter tuning methods highly benefit from being able to leverage them.
- **Maximal Number of Tuneable Hyperparameters:** Hyperparameter tuning methods must be able to handle the high numbers of hyperparameters that are demanded by complex ML-algorithms.
- **Robustness and Flexibility:** Hyperparameter tuning methods must be able to be applied to ML-algorithms in very different fields, on very different kinds of data and to find optimal configurations on hyperparameters of different types.

The rules in the knowledge base were created on past HPO-evaluations of scientific papers, cheat sheets, competitions (e.g. Kagggle, ChaLearn), past projects and knowledge elicitation. Retraining of the knowledge base is conducted upon HPO-technique performances of new use-cases. The ES maps the use-case description and information of data integration, data preparation as well as ML-algorithms with the rules, represented in the knowledge base. This allows a problem specific selection of an HPO-technique. For instance, for an ML-algorithm with purely real-valued hyperparameter configuration spaces and cheap evaluations of the objective function that allow more than hundreds of evaluations and a hardware infrastructure that allows parallelization, a population based method such as CMA-ES should be used.

4. VALIDATION AND EVALUATION

This chapter comprises the validation of the methodology for implementing ML and AI in manufacturing. Afterwards, the ES for selecting suitable HPO-techniques is evaluated.

4.1 Validation of Methodology for Implementing Machine Learning in Predictive Process Control

In the first step, the **AI-use-case selection** is performed. The selected use-case is the milling process of a blade integrated disk (Blisk). Blisks are used in jet engines in compressor stages, while requiring high specific strength, good corrosion as well as temperature resistance. The exact adherence of tolerances in terms of geometry and roughness is necessary to ensure highest efficiency in operation. For this reason, the achievement of high product quality through milling is mandatory. Since Blisk milling is a long-durational and expensive process, quality issues require long and costly rework or even scrap, which need to be avoided. One main quality issue during milling is the occurrence of chatter marks that are introduced when vibrations occur. Hence, it is promising to predict the vibrations and adjust parameters beforehand to avoid the vibrations. The ML-task is to predict future vibrations, which can be captured by the acquisition of the sound pressure level (SPL). In case of a performant ML-model, the process parameter can be adjusted via a machine controller. An exemplary Blisk can be seen in Figure 3 (Technicut).

Figure 3: Blade Integrated Disk

The first step of the ML-pipeline consists of data integration, since two different data sources are available. One data set comprises the information of the sound pressure level, and the second one the machine data such as spindle speed and axes. Because of different sample frequencies, the data sets need to be integrated as well as synchronized.

Afterwards, **data preparation** is carried out according to the structured data preprocessing pipeline. A data quality check is performed to identify necessary preprocessing techniques. The outcome of an initial data quality check reveals the necessity of data cleaning, since first entries are missing values. Secondly, the number of dimensions are
During the phase of feature engineering, the SPL of the current and past two observations are used for the creation of further attributes. Both first and second derivative of the signal is calculated to identify the slope and change of SPL. Since the model will be used for adjusting process parameters in advance, a prediction horizon H is calculated in order to determine, which time is necessary to correct corresponding variables. The prediction horizon H is set as H = 184.255 ms.

The selection of the ML-algorithm is the first step in the modelling phase. Based on requirements of successful ML-model implementations on similar use-cases, the milling process and computing power available, the ML-algorithms are selected. Since production experts require transparent ML-algorithms, random forest (RF) and gradient boosting (XGBoost) are chosen. The ML-algorithms are trained using Python and its open source libraries. Since the learning task of the use-case is regression, the model performance is assessed based on the metrics root mean square error (RMSE) and mean absolute error (MAE). The final RMSE-values of the most performant ML-model XGBoost is RMSE = 0.180. This result can be used for further improving the ML-pipeline by enhancing data quality.

Afterwards, a deployment strategy can be started. Since the information is applied for an adjustment of process parameters, the decision is automated. The model will be trained offline after a previously defined schedule and implemented into the machine controller. A continuous testing of the software system is ensured. If the model is exceeding RMSE-values higher than RMSE = 0.250, the model needs to be retrained. Lastly, the AI-system need to be certified, especially in the context of aerospace engineering. The certification process can be started after the ML-model is finally deployed.

4.2 Validation of ES for Selection of HPO-Techniques

The hyperparameter configuration space for XGBoost and RF was selected based on previous projects. Considering the use-case description, data set, boundaries, selected ML-algorithms and hardware settings, the ES recommends SMAC over other HPO-techniques. Random Search (RS), Gaussian Process (GP), Sequential Model-Based Algorithm Configuration (SMAC) as well as Tree Parzen Estimator (TPE) were implemented for the comparison for evaluation. RS was selected over GS (Grid Search), since GS would suffer from different search ranges and combinations of hyperparameters for both ML-algorithms due to exponential growth with the dimensionality of the configuration space. The validation loss curves with the median and variances of RS, GP, SMAC and TPE are highlighted in Figure 4.

The dashed line represents the XGBoost with default hyperparameters, while further lines show median and coloured area variances based on three rounds with each 50 iterations. The loss at the beginning is worse than the performance of the ML-model with default hyperparameters. However, final performances are improved within the first 20 iterations. After approximately iteration number 8, all HPO-methods prove better validation performances than the ML-model with default hyperparameters. From iteration 20 to 50, the improvement nearly stagnates and at the end of the 50th iteration, all HPO-methods present similar validation losses around RMSE = 0.31.

Figure 4: Loss Validation Curve for XGBoost

The validation loss curves for RF including mean and variances of RS, GP, SMAC and TPE and the performance of the RF with default hyperparameter settings for three rounds reveal similar results. The results show from beginning for each HPO-method better validation performances than the RF-model with default hyperparameter setting. In addition, GP, SMAC and TPE improve mainly the loss performance within the first 10 iterations, whereas RS present worse performances up to 20th /30th iteration. After iteration number 50, validation loss curves of all HPO-methods stagnate until iteration number 50. Ultimately, all HPO-methods exhibit similar validation losses at around RMSE = 0.306.

The RF-models and XGBoost-models with the respective best hyperparameter settings of RS, GP, SMAC and TPE are tested on an unseen test set. The performance is improved by all HPO-methods for both ML-algorithms. SMAC highlights best performance improvement with 2.6 % for XGBoost, whereas RS performs best with 9.8 % for RF. In general, all optimised RMSE values for XGBoost and RF exhibit similar results. However, compared to validation losses, the test RMSE values of RF-models and XGBoost-models are much lower. In summary, RS, GP, SMAC and TPE improve both XGBoost- and RF-models in validation as well as testing compared to default hyperparameter settings. The final optimised test results exhibit equivalent RMSE scores for XGBoost and RF with the trend for minimal better ones for RF. However, the XGBoost-algorithm is improved in the range of 1.1 % to 2.6 % and RF in the range of 9.4 % to 9.8 %.

Overall, SMAC provided the largest improvement in the performance for XGBoost, being outperformed by RS. One possible explanation for RS outperforming SMAC when using RF lies in the large amount of iterations and the similar improvements achieved for RF.

5. CONCLUSIONS

Many success stories already exist regarding the implementation of ML and AI in manufacturing. However, it is still challenging for companies to conduct AI-projects methodologically and to know how to optimize its pipeline.
For this reason, we introduced a concept how to proceed in an AI-project. The concept comprises the steps of selecting promising AI-use-cases, integrating and preparing data, modelling, deployment and certification. We validated this approach based on the milling process of Blisks. In this context, we focused especially on tuning hyperparameters as one very promising optimization potential within the ML-pipeline. For the selection of HPO-techniques we proposed the BRBES, which provides a recommendation on which HPO-technique to choose depending on the scenario. This ES can be used for the decision support of a data scientist, but also for warm starting a ML-system (initialization of the solution and reduction of search space). The structure of the ES can be transferred to other phases of the ML-pipeline such as data preparation. Further work in the context of ML will be focused on the automation of the whole ML-pipeline and introduction of ES for the other ML-pipeline phases. Regarding the validation use-case, the adjustments of parameters by the controller need to be strengthened.

6. ACKNOWLEDGEMENTS

The research in this paper was partially supported by the European Commission through the H2020 project EPIC (https://www.centre-epic.eu/) under grant No. 739592.

REFERENCES

AbdElRhahman ElSaid; Fatima El Jamiy; James Higgins; Brandon Wild; Travis Desell (2018): Optimizing long short-term memory recurrent neural networks using ant colony optimization to predict turbine engine vibration. In: Applied Soft Computing 73, S. 969–991. DOI: 10.1016/j.asoc.2018.09.013.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R.: CRISP-DM: Step-by-step data mining guide, https://www.the-modeling-agency.com/crisp-dm.pdf, 1999.

Cognilytica: CPMAI: Methodology for Implementing AI Projects Successfully. Online verfügbar unter https://www.cognilytica.com/cpmai-methodology/

Dong-Ling Xu; Jun Liu; Jian-Bo Yang; Guo-Ping Liu; Jin Wang; Ian Jenkinson; Jun Ren (2007): Inference and learning methodology of belief-rule-based expert system for pipeline leak detection. In: Expert Systems with Applications 32 (1), S. 103–113. DOI: 10.1016/j.eswa.2005.11.015.

Eric J. Horvitz; John S. Breese; Max Henrion (1988): Decision theory in expert systems and artificial intelligence. In: International Journal of Approximate Reasoning 2 (3), S. 247–302. DOI: 10.1016/0888-613X(88)90120-X.

Feurer, Matthias; Hutter, Frank (2019): Hyperparameter Optimization. In: Frank Hutter, Lars Kotthoff und Joaquin Vanschoren (Hg.): Automated Machine Learning: Methods, Systems, Challenges. Cham: Springer International Publishing, S. 3–38.

Hutter, Frank; Hoos, Holger H.; Leyton-Brown, Kevin (2011): Sequential Model-Based Optimization for General Algorithm Configuration. In: Carlos A. Coello Coello (Hg.): Learning and Intelligent Optimization. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 507–523.

James Bergstra; Yoshua Bengio (2012): Random Search for Hyper-Parameter Optimization. In: Journal of Machine Learning Research 13 (10), S. 281–305. Online verfügbar unter http://jmlr.org/papers/v13/bergstra12a.html.

Jian-Bo Yang; Jun Liu; Jin Wang; How-Sing Sii; Hong-Wei Wang (2006): Belief rule-base inference methodology using the evidential reasoning Approach-RIMER. In: IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 36 (2), S. 266–285. DOI: 10.1109/TSMCA.2005.851270.

Krauß, J.; Frye, M.; Döhler Beck, G. T.; Schmitt, R. H. (2019a): Selection and Application of Machine Learning-Algorithms in Production Quality. In: Machine Learning for Cyber Physical Systems, Bd. 9, S. 46–57.

Krauß, Jonathan; Dorißen, Jonas; Mende, Hendrik; Frye, Maik; Schmitt, Robert H. (2019b): Machine Learning and Artificial Intelligence in Production: Application Areas and Publicly Available Data Sets. In: Jens Peter Wulfsberg, Wolfgang Hintze und Bernd-Arno Behrens (Hg.): Production at the leading edge of technology. Springer Berlin Heidelberg, S. 493–501.

Leilei Chang; Yu Zhou; Jiang Jiang; Mengjun Li; Xiaohang Zhang (2013): Structure learning for belief rule base expert system: A comparative study. In: Knowledge-Based Systems 39, S. 159–172. DOI: 10.1016/j.knosys.2012.10.016.

Marc Claesen; Bart De Moor (2015): Hyperparameter Search in Machine Learning. In: CoRR abs/1502.02127.

Microsoft Azure (2020): Was ist der Team Data Science-Prozess (TDSP)? Online verfügbar unter https://docs.microsoft.com/de-de/azure/machine-learning/team-data-science-process/overview

Ron Kohavi (1995): A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: MORGAN KAUFMANN, S. 1137–1143.

Stack Overflow (2017): sklearn: Hyperparameter tuning by gradient descent? Online verfügbar unter https://stackoverflow.com/questions/43420493/sklearn-hyperparameter-tuning-by-gradient-descent.

Swati Sachan; Jian-Bo Yang; Dong-Ling Xu; David Eraso Benavides; Yang Li (2020): An explainable AI decision-support-system to automate loan underwriting. In: Expert Systems with Applications 144, S. 113100. DOI: 10.1016/j.eswa.2019.113100.

Technicnet: BLISK. Online verfügbar unter https://www.technicnet.co.uk/blisk/?page=blisk

Thornton, C.; Hutter, F.; Hoos, H. H.; Leyton-Brown, K. (2013): Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In: Proc. of KDD-2013, S. 847–855.

Wang-Chi Cheung; Weiwen Zhang; Yong Liu; Feng Yang; Rick-Siow-Mong Goh (2018): Automated Hyperparameter Tuning for Machine Learning Models in Machine Health Prognostics.

Xu, Dong-Ling (2012): An introduction and survey of the evidential reasoning approach for multiple criteria decision analysis. In: Annals of Operations Research 195 (1), S. 163–187. DOI: 10.1007/s10479-011-0945-9.