A data-based multi-algorithm system for an end-to-end intelligent manufacturing process

Artur Freitas Gonçalves (afreitas@fe.up.pt)
Universidade do Porto Faculdade de Engenharia https://orcid.org/0000-0003-2355-990X
João Reis
Universidade do Porto Faculdade de Engenharia
Gil Gonçalves
Universidade do Porto Faculdade de Engenharia

Research Article

Keywords: Optimization, Decision support systems, Predictive models, Genetic algorithms, Zero-shot Learning, Wood industry, Explainable AI

Posted Date: August 5th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-735806/v1

The International Journal of Advanced Manufacturing Technology manuscript No.
(will be inserted by the editor)

A data-based multi-algorithm system for an end-to-end intelligent manufacturing process

Artur Freitas Gonçalves[1] · João Reis[1] · Gil Gonçalves[1]

Received: date / Accepted: date

Abstract Reducing defects and creating new industrial products are issues addressed in literature, though often separately. This work presents a unique approach combining machine learning and optimization techniques, which could 1) predict and explain ongoing production defects, 2) prescribe solutions to prevent predicted defects, and 3) produce defect predictors for novel products without historical data. By applying it into a melamine-surfaced boards process, task 1) explored SVM, XGBoost and Random Forest algorithms, coupled with an explainable model-agnostic approach. A Powell's method-based meta-heuristic algorithm handled task 2), while the Hyper-Process Model (HPM) technique was selected in task 3). Defect prediction presented strong results, with a fine-tuned XGBoost with oversampling achieving over 0.8 recall value, and prescriptions significantly reduced prediction scores across most defect scenarios. Although the HPM implementation introduced challenges producing predictors, most generated synthetic data-trained models achieved encouraging performances on recall metrics. Results could serve as baseline for future developments.

Keywords Optimization · Decision support systems · Predictive models · Genetic algorithms · Zero-shot Learning · Wood industry · Explainable AI

Artur Freitas Gonçalves
E-mail: afreitas@fe.up.pt

João Reis
E-mail: jpcreis@fe.up.pt

Gil Gonçalves
E-mail: gil@fe.up.pt

[1] Research Center for Systems and Technologies (SYSTEC), Faculty of Engineering, University of Porto, 4200-465 Porto, Portugal

1 Introduction

Wood industry has been the topic for Industry 4.0 implementations across its value chain [1]. Melamine surfaced board is among the products offered by the industry, after a complex multi-step manufacturing process. Paper is impregnated with melamine resin, and coated into a raw board surface via a hot-pressing process. This process requires handling complex relationships, including pressure and temperature manipulation, as well as recipes development [2–4].

Due to technological, scientific, and human factors, Artificial Intelligence (AI) has been a hot topic in recent years. Although significant advancements were made, open problems remain yet to be solved in Academia and Industry [5]. Among them, the resource-intensive nature of many machine learning applications could not only present a barrier to entry for late players, but also be a sustainability concern for its carbon footprint [6–8]. Optimizing melamine-surfaced boards production could be achieved by exploring advanced analytics and AI techniques. Common practice has been partitioning production optimization problems into multiple subtasks, and address one at a time. Image classification tasks based on Deep Learning architectures were suggested for product identification [9], or defect detection [10–12]. Other works developed optimization solutions for wood stock sizing [13,14], or using the wood's physical and chemical properties to generate new products [15]. Material procurement and consumption optimization were suggested to tackle waste reduction [16].

This work introduces a set of machine learning and optimization techniques combined into a unique approach, capable of solving three production-related subtasks. The proposed multi-layered model is devised to: 1) predict a defect in real-time, and explain its occur-

rence; 2) recommend a real-time recipe to avoid it; and 3) predict defects for never-seen-before products. An extension upon CRISP-DM methodology is also proposed to construct and iterate over the three subtasks.

On section 2, the concepts and terminologies explored in this work are introduced, along with analogous or similar works available in literature; section 3 elaborates on the methodological structure adopted to enable constructing and maintaining the three main subtasks of the model; section 4 details modelling implementation for each subtask, including description of selected models and algorithms; section 5 presents overall and subtask specific results and performance analysis, which are further reviewed in section 6. This section also explores future work and development.

2 Literature Review

2.1 Zero-shot Learning

Humans are inherently capable of identifying and classifying thousands of objects after being taught their definition and looking over a small set of similarly looking objects. Unlike humans, most AI methods and techniques require a massive collection of samples to train state-of-the-art algorithms [17,18].

Transfer learning is a technique used to transfer knowledge obtained from a task to a slightly different new one [19]. It intends to mimic the human process of knowledge transfer which efficiently translates past experiences into new tasks [20]. It encompasses several approaches which focus on efficient usage of available data. Few-shot Learning (FSL) aims to identify new classes by using a very small sample size per class, while one-shot Learning (OSL) attempts to achieve the same by using only one sample per class [7,21,22]. More recently, a "less than one-shot Learning" approach was proposed, where less than one sample per class is used – that is, in order to learn N classes, $M < N$ samples are used [23].

Zero-shot Learning (ZSL) is an extreme approach for cases in which new tasks are identified without any available training data, other than a description of the classes or tasks themselves [24]. ZSL work has been presented in various tasks, including image processing for classification [25], image-text embedding models [26], or reinforcement learning [27]. This work will use a ZSL algorithm for the novel product development task named Hyper-Process Model (HPM), previously used in a regression task for Laser Seam Welding process optimization [28].

2.2 Metaheuristics-based Search

Metaheuristics-based search is a technique used to address combinatorial optimization problems. It is typically designed to find the best possible solution within a reasonable time by using approximate search methods [29,30]. It contrasts with exact optimization techniques, which often seek a problem's optimum solution by exploring cost function properties or the full spectrum of feasible solutions. Combinations of both approaches have been suggested in recent years, often being named hybrid methods [30].

Solution search may be associated with costly time and computational efforts, and literature finds that a trade-off between performance and solution quality is often better when searching within a single neighborhood, instead of searching in the neighborhood of every feasible solution [31]. Local search focuses on finding solutions within the neighborhood of a point. Powell method is suited for local minimum search in single objective functions with multiple variables, such as the scenario being implemented in this work, and has been used in literature to support enhanced solutions in continuous optimization tasks [32].

2.3 Explainable Artificial Intelligence

Although Explainable Artificial Intelligence (XAI) traces its origins back to earlier knowledge-based AI expert systems, it has become an active research area in recent years [33,34]. As AI systems are implemented with increasing complexity in a wide rage of tasks, understanding and interpreting results become more difficult. AI architectures such as Deep Neural Networks may assemble millions of parameters across hundreds of layers, whose performance renders it being known as a black box model [34]. Lack of transparency and interpretability are major liabilities for AI implementation in several fields of expertise, such as healthcare and finance, where accountability, fairness and trust are considered mandatory requirements [34,35].

Multiple approaches are found in literature to address this issue. Those algorithms whose explainability and understanding may be achieved using the algorithms themselves are grouped in a family of approaches named integrated interpretability - among them, include linear or logistic regression, decision trees, or K-Nearest Neighbors. Other algorithms, however, require post-hoc methods if explainability and transparency are implementation requirements. Neural Networks, Support Vector Machines and Tree Ensembles are examples of complex algorithm which require further work to reduce their opaqueness [34,35]. Post-hoc methods

may be further divided between: model-agnostic, when they only work with the model's inputs and outputs; or model-specific if they use additional information from the model. An example of a model-specific method is when the learning information contained in a single decision tree is used to infer explainable reasoning from a decision tree ensemble it belongs to [35]. This work will explore a model-agnostic method with the EVADE algorithm, whose sole known implementation focused on interpreting a fraud detection algorithm [36].

3 Methodology

Although the multi-layered model incorporates several typically independent algorithms - each one designed to address a specific subtask -, their symbiotic nature demands for a slightly modified version of the CRISP-DM methodology [37,38]. This methodology is focused on data mining tasks, whose steps we found limiting the full toolkit required for this project. Thus, a circular, end-to-end solution connecting the various subtasks and algorithms was built. The full process comprised on setting an innovative, unified algorithmic recipe to 1.1) predict the occurrence of defective samples, 1.2) and explain it; 2) suggest a real-time recipe recommendation to avoid predicted defects; 3) and build a defect prediction model for a novel, previously unseen product. While subtask 1.1) relied on standard CRISP-DM, the remaining subtasks required adaptations, as nearly every algorithm impacts each other's performance. This process rendered our methodology into a more general, guiding approach for other AI projects which may not solely rely on standard CRISP-DM tasks (Figure 1).

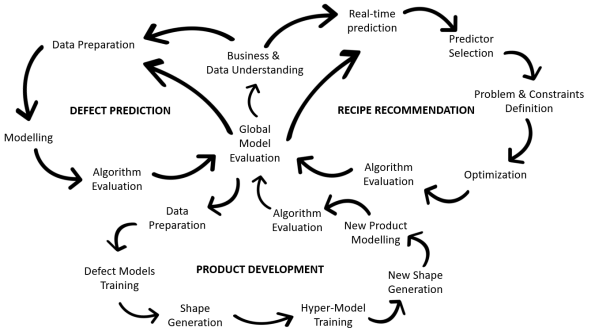

Fig. 1 Symbiotic CRISP-DM methodology.

3.1 Defect Prediction Modelling

Traditional statistical methods were applied in order to improve knowledge on the business and its available data [37]. The main goals were to understand relationships between features and tasks, recognize outlier data points and noise, and extract new features. Correlation statistics and dimensionality reduction were used to handle data issues, such as anomalies, redundancy, or the volume of data itself.

Several established algorithms could be used to address the classification task - e.g. Random Forests, Gradient Boosting Classifier, and Support-Vector Machines -, with distinct strategy implementations: using default implementation parameters; hyperparameter optimization search; oversampling techniques, among others. As defects in manufacturing have an outlier nature, it was likely that class imbalance would be seen in data. When defect detection is the primary target, it might become relevant to consider model performance metrics such as recall, over precision and F1-Score.

An XAI algorithm may enable explainable insights on which features are most effective on real-time defect classification inferences. While some explainable algorithms are available to work on specific modelling algorithms, other models allow for model-agnostic explainability. Such type of works could allow explaining why a sample would be likely to produce a defect, regardless of the underlying prediction modelling process. In this work, the explainable implementation is built upon a model-agnostic optimization procedure named EVADE. It works directly with the real-time sample, its defect prediction score, and feeds from a subset filtered by that score's defect type. Instead of generating a genetic dataset as the original EVADE algorithm, we will use the testing subset from defect prediction modelling [36].

3.2 Real-time Recipe Recommendation

In an industrial setting, a recipe is a set of parameter values combined to manufacture a product. Although unlikely, recipe changes might be necessary if a defect is likely to occur. A metaheuristics-based search approach may be adopted to generate a real-time recipe recommendation and reduce a sample's defect prediction score. The recipe recommendation algorithm produces value recommendations for key features which may be manipulated under a very short time span, and whose identification requires domain knowledge. A generic cost function may be used, but real-time scenarios require feature manipulation restrictions in short time spans. As these restrictions reduce the available search space for optimization, it is likely that only local minima remain available, rather than the global minimum. To ensure the recommended recipe's feasibility,

boundaries were set so that no feature change is recommended beyond defined limits. A local search-based method with boundaries, such as the Powell method, could be an optimal approach for this task.

3.3 Novel Product Development

On product development, the ultimate goal is to propose the recommendation of a novel product with predetermined values on key features, while avoiding defects. Although it might be an identical ontology as the one previously conceptualized in defect prediction and recipe recommendation processes, the methodological steps were quite different here, as no data was yet available for that new product. This work explores a ZSL approach to overcome that limitation.

The Hyper-Process Model (HPM) algorithm is capable of producing a defect prediction model with no new data. It takes the new task's key features values and pre-trained models for similar tasks as inputs, and generates the required data to train a new model. In this case, defect prediction models - or "predictors" - are trained to identify defects on pre-existing products. Key statistical data is compiled from each predictor to form a "shape", that is, a synthetic dataset labelled by that predictor. The term "shape" is borrowed from the Statistical Shape Model (SSM) technique used in the HPM. The shapes are then used, along with the key features values from the unseen task, to generate and train the HPM algorithm. This model predicts a new shape - a synthetic labelled dataset - based on the specific characteristics of the novel product. At last, the new shape is used to train a defect prediction model for the never-seen-before product [28].

4 Implementation

Python language applications were used to implement every methodological step presented above, including tools such as Pandas, NumPy, scikit-learn and XGBoost. A similar implementation of this work was recreated on Microsoft Azure ML's block-based interface to support this work's real-case deployment.

4.1 Business and Data Understanding

Melamine-surfaced boards production consists on a process divided into six stages: resin preparation; paper preparation; bath and drying; paper cooling and cutting; table incorporation; and storage. Products considered in this work were based on the particle board and the medium-density fiberboard (MDF) types.

Resin preparation step encompasses its recipe elaboration process, whose ingredients are collected and mixed under controlled temperature. Data collected onsite included temperature, humidity, tanks usage, as well as critical recipe formulation ratios prepared by domain experts. Paper preparation step relates with logistics data impacting production orders. It included batch size and quantity, paper physical properties, and shelf life. Bath and drying step consisted in impregnating paper inside a tank with melamine resin. Coating and straightening are performed during the impregnation process. Following impregnation, the paper is dried up and stored. At this stage, environment data (e.g., temperature and humidity), as well as machine, paper and melamine-related data were collected. After a cooling and cutting step, the table incorporation is performed. Here, impregnated paper is pressed against the wooden board by a pair of plates, before storage and transportation. Data collected here included pressing machine performance, and periodical end-of-the-line quality control.

Available data spanned the entire production process, and was either collected from sensors and machines spread across multiple production lines, from historical databases, or manually created after production. It ranged from September 2018 to May 2020, spanning 597 days. For defect prediction modelling tasks, production samples were labelled by domain experts on whether they were normal or defect samples. Defects were further divided into one of nine defect types, ranging from paper problems, melamine related anomalies, to logistics or environment issues. The most frequent and high priority defect type, "Broken Paper", was the sole defect type explored in defect prediction modelling, explainable algorithm, and recipe recommendation optimization processes. On novel product development process, all defect subcategories were considered and combined into a single, unified defect class. Datasets were instead filtered by their unique product subcategories.

4.2 Data Pre-Processing

Since multiple data sources were considered, the first step concerned merging all data into a single, unified dataset. Data containing the unique production order code per sample was merged by that code, while the remaining data was merged by time. A two-step approach was devised for the IoT data: first, data was merged by maximum time granularity possible, in seconds; then, samples were aggregated by minute, product subcategory and defect type. Whenever multiple samples shared the same minute timestamp after first

step, they would be aggregated by their mean value per feature.

Due to inconsistencies, limitations, or errors, the dataset required further pre-processing techniques before being suitable for modelling. Feature columns containing a large amount of missing data were removed, and one member was removed from any pair of features with Pearson correlation coefficient equal or over |0.9|.

At this stage, the dataset was separated into two major versions. While both kept normal samples, one version was prepared for the defect prediction modelling and recipe recommendation processes, where only "Broken Paper" defect was kept. The other version would feed the novel product development, for which every defect type was kept and combined into a single defect class. This version was then filtered to keep the 5 product subcategories with most samples.

Further analysis was performed to extract time series features from data. After visual exploratory analyses, a more robust and evenly distributed time frame was selected from October 1st, 2018 up to March 31st, 2020. It reduced inconsistencies or inaccuracy risks related with initial onsite implementation of IoT devices. Additionally, most production activity impacts caused by SARS-CoV-2 pandemic was avoided. Environment- and temperature-related features were then identified for time series feature extraction. An autocorrelation function (ACF) was performed to detect nonrandomness in data, using Python's statsmodels implementation [39,40]. Features with strong autocorrelation were subjected to a delta variation extraction for each consecutive sample, adjusted to time. This adjustment was made due to the fact that data was no longer uniformly distributed in time. Although time granularity was previously set as 1 minute per sample, a considerable number of samples was removed during data pre-processing. The procedure considerably reduced autocorrelation issues in data. An additional delta time feature was extracted, with the purpose of detailing time variation between consecutive samples.

The final dataset used in defect prediction modelling and recipe recommendation contained 53,300 samples and 33 features - including target. Nearly 3,000 samples were defective, corresponding to 4.9% of the dataset and confirming the classes' unbalanced nature. Pearson correlation coefficients now present the result of pre-processing data procedure, and displays a clean dataset, with low correlation and no missing data (Figure 2).

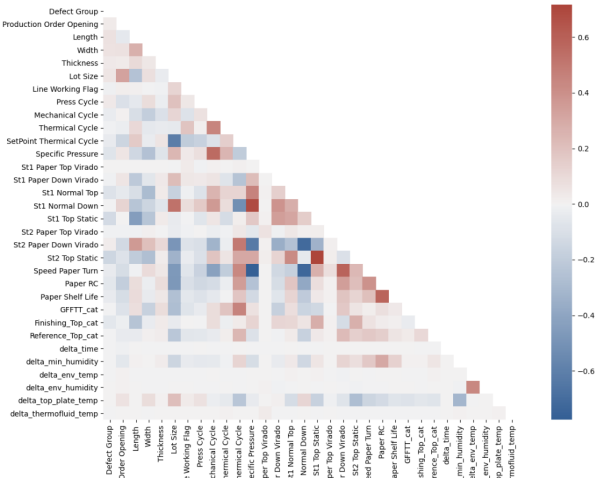

Fig. 2 Pearson correlation coefficients on clean dataset.

4.3 Defect Prediction Modelling

Since cleaned data was tabular, a performance comparison was done between the following algorithms: Random Forest, Support Vector Machine (SVM), and XGBoost. The first two were implemented using Python's scikit-learn library, while the latter was implemented with XGBoost library [41,42].

First, the data was transformed with the min-max normalization technique, except for the target feature. The purpose was to encapsulate all features within an identical range between [0,1] - without losing relationships among original values -, and achieve less training computational effort towards convergence, while minimizing overfitting risks. Since only one defect type was kept in this dataset version, each algorithm performed as a binary classifier, where: normal class was 0; defect class was 1. The dataset was subjected to a pseudo-randomized train-test sampling split, in which 80% of samples were exclusively used in model training, while the remaining 20% were saved for model testing.

Three distinct model optimization strategies were considered. First, default parameters were selected for each algorithm's implementation. Then, an oversampling technique was implemented to address class imbalance, while maintaining default parameters. The defect samples' amount in the training subset was increased 10-fold, which resulted in a increase to 29% of defect class representation. The third strategy combined oversampling with a hyperparameter optimization process, using scikit-learn's GridSearchCV. It comprised a heuristic-based search for an optimal combination of hyperparameters for each algorithm. The performance goal was to correctly recognize as many defect samples as possible. Thus, every model was trained to optimize recall metric. Still, precision and F1 Score metrics were calculated and analysed for each model.

4.4 Explainable Algorithm

The explainable algorithm required additional key input data to be collected. Features were sorted as containing categorical or numerical values, and each feature's cardinality was calculated - that is, the number of unique values per feature. EVADE algorithm uses a heuristics-based search to identify the real-time sample's counterfactual sample [36]. The latter corresponds to the most similar sample available in a dataset whose class is opposite to the real-time sample, and the prediction model correctly classified it.

The counterfactual sample is found by calculating similarity and fitness scores between the real-time sample and each sample in the dataset. After calculating Jaccard index for categorical features and Cosine similarity for numerical features, the similarity score was the weighted sum between the two values - in this case, identical weight was given. Fitness score calculates the prediction score discrepancy between the real-time sample and each data sample. Similarity and fitness scores were summed with equal weights, and the sample with highest final score was selected as the counterfactual sample. Since we wanted to attribute highest fitness score to samples whose defect score was furthest away from real-time sample's, a slight change was made to Equation (1).

$$f_{score}(X', p, t) = 1 - \frac{max(1-t, t) - |p(X') - t|}{max(1-t, t)} \quad (1)$$

In the final step, features with diverging value in both samples were stored. A heuristic approach was then used to calculate the total number of samples whose value differed from the real-time sample - a process repeated for each stored feature. The final importance value was computed by dividing that total number by the feature's cardinality. The higher the importance value, the likelier that feature could explain the predicted class. To evaluate the algorithm's performance, a batch of one thousand samples from testing subset were randomly selected, and each feature's mean relative importance score from EVADE was measured against Tree-SHAP, an explainable algorithm for tree ensemble methods [43].

4.5 Real-time Recipe Recommendation

The first step to build the recipe recommendation algorithm was retrieving the defect prediction modelling's testing subset. It was filtered to pick out the true negative sample with lowest defect prediction score, which would be the target optimization sample. Then, domain knowledge identified five features which could be changed during real-time production, that is, could be manipulated during optimization to reduce defect prediction scores.

The real-time recipe algorithm was metaheuristics-based, and its optimization step included the Powell method's scypi implementation, which admits bounded search [44]. The squared Euclidian distance between real-time sample and the true negative sample was selected as the minimization target. Its output is the recipe recommendation, which is expected to produce a lower defect score than the real-time sample. For evaluation purposes, five sample groups were extracted from training subset to estimate the algorithm's performance against distinct levels of defect prediction scores.

4.6 Novel Product Development

Novel Product Development process was based on a solution available in literature named Hyper-Process Model (HPM), focused on solving regression tasks [28]. This technique is used to overcome the absence of real data to train a new defect prediction model, assuming that other predictors exist for similar, yet different, tasks.

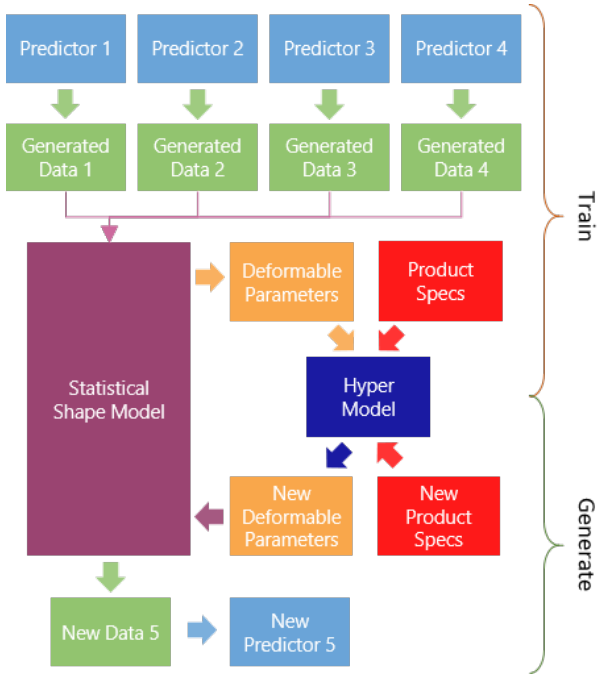

Fig. 3 Hyper-Process Model flow process for training and predictor generation.

As presented in Figure 3, the predictors were first trained for defect detection in each product, by emulat-

ing the best performing strategy from subsection 4.3. Four of the predictors would serve as training sets for the HPM, while the remaining one would be the validation set. This is part of a 5-fold cross-validation technique to generate different combinations of four training sets and one validation set, so that the HPM's performance is evaluated. The predictors generate the initial shapes, which comprise the generated data used in training (Figure 3).

The idea behind these shapes is to assess how each predictor varies its output according to a specific problem - in this case, a product. To perform such a variational analysis, the creation of each shape is based on synthetic dataset generated to feed as input each predictor. This means that each predictor's input have the same number of samples, same distribution of values and same features. This synthetic dataset is required to be the same for each predictor where a shape is constituted solely by predictor's output.

By setting its number of unique data points to 1.5×10^6, the synthetic dataset was generated using a Stochastic Factorial Design (SFD), which introduces random non-linearity in sample values generation, in order to increase the probability of some samples being defective within each feature's min-max values. Techniques based on linear generation of data would require an extremely higher number of samples to grasp the defect distribution, hence the stochastic approach. Each predictor would annotate the resulting 1.5×10^6 dataset, generating a shape. Since input dataset is identical for every predictor, differences among shapes are assessed through their annotations. In order to improve class balance, an additional heuristic approach was performed to improve the distribution ratio by removing mostly samples which were part of a dominant class. The process resulted in a 8,000 samples dataset, where 2 out of the 5 products had nearly 50% class representation and another product had about 20% defect samples. Regarding HPM's hyperparameters, its polynomial degree was set to 4 – the number of training sets.

The four predictors' shapes were then flattened and stacked on top of each other, with each shape pairing up with the product's six key specifications. The stacked matrix was first decomposed using the Statistical Shape Model (SSM) method, where a set of deformable parameters is produced per shape. These parameters may be interpreted as a latent space which allows the reconstruction of each shape. The HPM is trained by using the deformable parameters, along with the products' specifications.

Upon novel product's specifications, the HPM predicted its deformable parameters, which were used by the SSM to generate the new product's shape. This shape served as the dataset to train the new predictor. Its recall, precision and F1-Score metrics were collected and compared against the predictor trained with original data from the target product. Every performance metric was obtained by using the validation set's testing data.

5 Results

Modelling results with different combinations of algorithm and optimization strategy for the task of predicting "Broken Paper" defect type may be observed in Table 1. The implementation presenting highest recall value was the XGBoost (XGB) algorithm with oversampling (over.) and hyperparameter optimization (optim.). Regardless its implementation, XGB clearly outperforms the other two algorithms, as the algorithm with second highest recall value is Random Forest (RF) with oversampling and hyperparameter optimization - which performs 8 percentage points lower in recall metric. Analyzing the other two performance metrics, the model with highest precision was RF with default parameters. Regarding F1 Score, the model having the best performance in that metric was the XGB using default parameters.

Table 1 Prediction Modelling Results

Model	Precision	Recall	F1-Score
RF	0.926	0.707	0.802
XGB	0.910	0.742	0.817
RF + over.	0.890	0.726	0.800
RF + over. + optim.	0.900	0.732	0.807
XGB + over.	0.831	0.790	0.810
XGB + over. + optim.	0.712	0.812	0.759
SVM + over. + optim.	0.266	0.581	0.365

Overall, XGB presented highest recall values in every implementation type, with its implementations using default parameters or oversampling both scored the highest F1 Score among all algorithms. RF performed best in precision metrics, with its implementation using oversampling and hyperparameter optimization outperforming every other algorithm's F1 Score. SVM obtained the lowest value on every metric. It should be noted that its sole displayed implementation was with oversampling and hyperparameter optimization, as results for every other strategy was close or equal to zero. SVM's best results were obtained using the polynomial kernel.

From the test subset used to evaluate the defect prediction models, one thousand samples were randomly

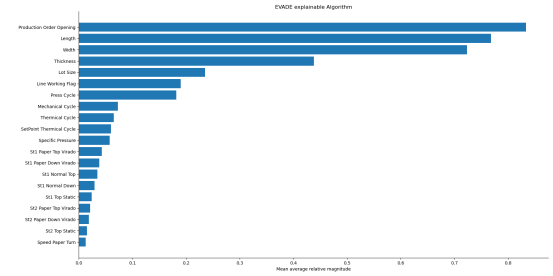

Fig. 4 Features' mean importance magnitude with EVADE algorithm.

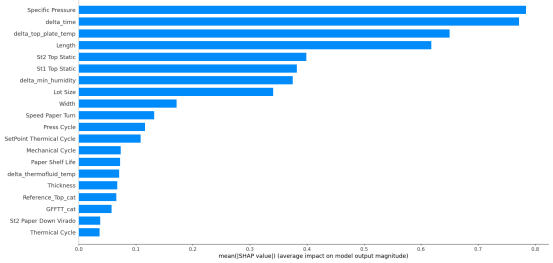

Fig. 5 Features' mean importance magnitude with Tree-SHAP algorithm.

selected to analyse our explainable algorithm's performance, EVADE. On Figures 4 and 5, it listed the top-20 features whose importance explaining defect prediction scores is highest, according to EVADE and Tree-SHAP algorithms. EVADE algorithm highlights size and production cycle-related features - including features impossible to be modified in real-time, such as production order opening, length and width - as those impacting more how samples are classified by prediction models. On the other hand, Tree-SHAP presents time-series extracted features and size-related features, including specific pressure, delta time and top plate temperature variations, as the ones being more effective in the prediction task. Although specific pressure and top plate temperature's values were changeable in real-time, the remaining three features which could be manipulated in real-time were all outside the Tree-SHAP's top-10.

On recipe recommendation algorithm, five distinct levels of defect score were identified (Table 2). Results were widely positive, as average defect scores decreased 0.2 percent point (pp) across all subgroups after the recipe recommendations. It should be noted that the goal was to assess samples classified as defect (label 1), and suggest a non-defective recipe (label 0). Thus, it is key that lower values are achieved in *Avg. After* column. Subgroups [0.50, 0.75[and [0.75, 0.99[had their average score after recommendation below 0.5. Percentage change was over 0.5 across every subgroup, except [0.99, 1]. This subgroup corresponded to the more extremes cases of defect prediction, and the algorithm obtained lowest improvements, with a reduction of 0.239 percentage change to 0.756 average score.

Table 2 Recipe Recommendation's impact on defect prediction score

Original Score	Avg. Before	Avg. After	Diff pp	Pct. Change
[0.99, 1]				
Powell	0.994	0.756	0.238	0.239
Nelder-Mead	0.994	0.764	0.230	0.231
L-BFGS-B	0.994	0.771	0.223	0.224
[0.75, 0.99[				
Powell	0.939	0.359	0.580	0.618
Nelder-Mead	0.939	0.375	0.564	0.600
L-BFGS-B	0.939	0.446	0.493	0.525
[0.50, 0.75[				
Powell	0.605	0.081	0.523	0.866
Nelder-Mead	0.605	0.104	0.501	0.828
L-BFGS-B	0.605	0.259	0.346	0.572
[0.25, 0.50[				
Powell	0.341	0.071	0.270	0.793
Nelder-Mead	0.341	0.086	0.255	0.749
L-BFGS-B	0.341	0.185	0.156	0.457
False Positives				
Powell	0.261	0.064	0.198	0.756
Nelder-Mead	0.261	0.082	0.180	0.688
L-BFGS-B	0.261	0.174	0.088	0.335

Table 3 HPM's Training & Validation Sets Performance

Model	Precision	Recall	F1-Score
Product 1	0.863	0.808	0.835
Product 2	0.738	0.698	0.717
Product 3	0.176	1.000	0.300
Product 4	0.759	0.595	0.667
Product 5	0.600	0.667	0.632

Table 3 displays the defect prediction modelling performance per product, where the different defects are predicted in a multi-class manner. These will then serve as input for HPM (model of models). Applying a 5-fold cross-validation technique, four of those models were then used as HPM's training sets, while the remaining model was the validation set. Despite having the highest recall out of the five models, product 3 had the lowest precision and F1-Scores which could hinder HPM's training. The other models had more balanced out results, with product 1 presenting highest values across all remaining metrics, and a recall of 0.808.

Table 4 displays the performance from each predictor trained with HPM's generated synthetic data by only using 1) previously trained models and 2) new products characteristics. No data was used for the product left out of the 5-fold CV. The real testing subset

Table 4 HPM's New Models Performance

Model	Precision	Recall	F1-Score
Product 1	0.309	0.599	0.408
Product 2	0.289	0.931	0.441
Product 3	0.028	0.583	0.054
Product 4	0.124	0.514	0.200
Product 5	0.112	0.778	0.195

from Table 3 was again used to evaluate predictors performance. Recall metric values were all above 0.5, with product 2 reaching 0.931 and product 5 attaining 0.778 values. Though it was not the metric being optimized during model training, precision values were generally low and only product 1 and 2 reached values over 0.2.

6 Discussion

This work comprised a symbiotic algorithmic recipe to address multiple production optimization tasks. Any task's output quality could impact the remaining structure, as their output might serve as input in subsequent tasks. For example, new data impacts the defect predictor, which in turn impacts explainability, recipe recommendation and predictor's generation. Thus, extra effort was put on data preparation and defect prediction modelling, so that we could increase quality in later algorithms. As observed in Tables 1 and 3, validation results from the multiple defect prediction models presented robust scores, with best performing models achieving recall values over 0.8. Still, further data collection could lead towards stronger results.

The symbiotic nature is best observed in the dynamics between defect prediction modelling, explainability, and the recipe recommendation algorithm. Indeed, when the recipe recommendation has limited improvement on defect score for extreme scenarios - subgroup [0.99, 1] -, one could wonder if Powell optimization method would be underperforming. However, tests made with other methods - Nelder-Mead, and L-BFGS-B - produced worse results. This lead us to the possibility that data might be the limiting factor.

By analyzing EVADE explainable insights, one may observe that none of the adjustable features that compose a recipe are among the top-8. Even though Tree-SHAP presents a top-3 with adjustable features, several non-adjustable ones also stand out. Also, the remaining adjustable features contain limited explainable impact in Tree-SHAP analysis. An in-depth comparison between samples performing well or not in subgroup [0.99, 1] pointed to patterns on several non-adjustable features which could hint that we are observing extreme samples far from the defect decision boundary, which could mean outlier or miscollected data. Increased data collection could improve the subgroup's representation, enhance predictions quality, and eventually assist the recipe recommendation algorithm. Another strategy include exploring the explainability results and develop an optimization process particularly focusing on those adjustable features presenting higher potential impact.

Though it was not the focus of this work, a discrepancy is observable between EVADE and Tree-SHAP results, as the algorithms' top-10 features in explainable magnitude only have three common entries. The more puzzling inference is that both algorithms offered useful insights in the previous analysis on the recipe recommendation algorithm. Thus, further study should be done on EVADE algorithm, particularly concerning parameters definition and aggregated results production.

Per its original set up, the HPM's architecture handles regression problems. Initially, the task was to adapt it as-is into a two-class problem solver. However, results were not satisfactory when it came to construct the synthetic dataset, as most versions barely produced any defect-labelled sample by any pre-trained model. This issue was overcome by constructing the Stochastic Factorial Design, and then improving the class balance of defect samples across all five products towards a 50/50 equilibrium. The new models generated from the synthetic dataset with HPM's class annotations presented encouraging results on the recall metric.

Nonetheless, three opportunities for improvement in HPM were identified. First, the heuristic approach to improve defect samples representation in the synthetic dataset should ensure that no significant change is made to sample distribution, as it could hinder HPM's learning capabilities and the final defect prediction model quality. Second, our work used a simple activation function to convert HPM's original regression output into a binary one - any non-zero output value now represented the defect class. This approach had promising results, and recognized most defect samples for product 1. Still, it did not achieve identical performance for every product. Third, HPM's current architecture presents scalability limitations. In case our synthetic dataset is larger than traditional machine learning datasets, it could become overwhelming for the Principal Component Analysis step, which is HPM's in-built dimensionality reduction technique. For future works, if larger sets are explored, the suitability of other methods available in literature should be analyzed.

7 Conclusion

This work intended to introduce a novel approach to optimize a melamine-surfaced boards production. By uniquely combining an ensemble of machine learning

and optimization techniques, a pipeline was elaborated to predict a defect in real-time, produce an explanation for its occurrence, present a recipe recommendation to avoid it, and generate defect prediction models for never-seen-before products. An innovative methodology was developed to seamlessly integrate every process. Promising results were obtained by the various algorithms, and a symbiotic nature between them was recognized. Further investigation paths were identified, which could lead towards improved and more robust results.

8 Declarations

Funding

This work has received funding from European Union's Horizon 2020 research and innovation programme under the grant agreement EIT/EIT Manufacturing/SGA 2020/1.

Conflict of interest

The authors declare that they have no conflict of interest.

Availability of data and material

The data used in this work is under confidentiality agreement, thus is not available for public dissemination.

Code availability

The code used in this work is under confidentiality agreement, thus is not available for public dissemination.

References

1. F. Müller, D. Jaeger, M. Hanewinkel, Computers and Electronics in Agriculture **162**, 206 (2019)
2. X. Le Fur, M. Galhac, M. Zanetti, A. Pizzi, Holz als Roh- und Werkstoff **62**(6), 419 (2004)
3. Y. Liu, X. Zhu, Construction and Building Materials **66**, 132 (2014)
4. G. Nemli, M. Usta, Building and Environment **39**(5), 567 (2004)
5. M. Haenlein, A. Kaplan, California management review **61**(4), 5 (2019)
6. I.M. Cockburn, R. Henderson, S. Stern, The impact of artificial intelligence on innovation. Tech. rep., National bureau of economic research (2018)
7. B.M. Lake, R. Salakhutdinov, J.B. Tenenbaum, Science **350**(6266), 1332 (2015)
8. R. Nishant, M. Kennedy, J. Corbett, International Journal of Information Management **53**, 102104 (2020)
9. B. Siregar, U. Andayani, N. Fatihah, L. Hakim, F. Fahmi, in *Journal of Physics: Conference Series*, vol. 801 (IOP Publishing, 2017), vol. 801, p. 012051
10. S. Jung, Y. Tsai, W. Chiu, J.S. Hu, C.T. Sun, in *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)* (IEEE, 2018), pp. 1456–1461
11. R. Ren, T. Hung, K.C. Tan, IEEE transactions on cybernetics **48**(3), 929 (2017)
12. J. Shi, Z. Li, T. Zhu, D. Wang, C. Ni, Sensors **20**(16), 4398 (2020)
13. R. Morabito, L. Belluzzo, European Journal of Operational Research **183**(3), 1405 (2007)
14. A. Zanarini, in *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (Springer, 2017), pp. 293–301
15. Y. Chen, Z. Wu, Journal of Intelligent & Fuzzy Systems **35**(3), 2741 (2018)
16. J. Mendes, Ó. Oliveira, C.S. Pereira, P. Fernandes, in *New Contributions in Information Systems and Technologies* (Springer, 2015), pp. 585–591
17. Y. Bengio, I. Goodfellow, A. Courville, *Deep learning*, vol. 1 (MIT press Massachusetts, USA:, 2017)
18. I. Biederman, Psychological review **94**(2), 115 (1987)
19. S.J. Pan, Q. Yang, IEEE Transactions on knowledge and data engineering **22**(10), 1345 (2009)
20. L. Torrey, J. Shavlik, in *International Conference on Inductive Logic Programming* (Springer, 2009), pp. 234–248
21. L. Fei-Fei, R. Fergus, P. Perona, IEEE transactions on pattern analysis and machine intelligence **28**(4), 594 (2006)
22. J. Snell, K. Swersky, R.S. Zemel, arXiv preprint arXiv:1703.05175 (2017)
23. I. Sucholutsky, M. Schonlau, arXiv preprint arXiv:2009.08449 (2020)
24. H. Larochelle, D. Erhan, Y. Bengio, in *AAAI*, vol. 1 (2008), vol. 1, p. 3
25. J. Ghosn, Y. Bengio, IEEE Transactions on Neural Networks **14**(4), 748 (2003)
26. A. Frome, G. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, T. Mikolov, (2013)
27. J. Sinapov, S. Narvekar, M. Leonetti, P. Stone, in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (2015), pp. 725–733
28. J. Reis, G. Gonçalves, N. Link, in *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society* (IEEE, 2017), pp. 3396–3402
29. I. Ardiyanto, S. Sulistyo, et al., in *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)* (IEEE, 2018), pp. 545–550
30. E.A. Lemamou, P. Galinier, S. Chamberland, IEEE/ACM Transactions on Networking **24**(1), 137 (2014)
31. J.L. Lin, H.C. Chuan, Y.H. Tsai, C.W. Cho, in *2013 7th Asia Modelling Symposium* (IEEE, 2013), pp. 61–64
32. N.Q. Tuan, T.D. Hoang, H.T. Thanh Binh, in *2018 IEEE Congress on Evolutionary Computation (CEC)* (2018), pp. 1–8. DOI 10.1109/CEC.2018.8477860
33. R. Confalonieri, L. Coba, B. Wagner, T.R. Besold, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **11**(1), e1391 (2021)

34. A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Information Fusion **58**, 82 (2020). DOI https://doi.org/10.1016/j.inffus.2019.12.012
35. F.K. Došilović, M. Brčić, N. Hlupić, in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (2018), pp. 0210–0215. DOI 10.23919/MIPRO.2018.8400040
36. F. Piccinini, (2020)
37. C. Shearer, Journal of data warehousing **5**(4), 13 (2000)
38. G. Piatetsky-Shapiro, Online verfügbar unter http://www.kdnuggets.com/polls/2014/analytics-data-mining-data-science-methodology.html (2014)
39. G.E. Box, G.M. Jenkins, G.C. Reinsel, G.M. Ljung, *Time series analysis: forecasting and control* (John Wiley & Sons, 2015)
40. S. Seabold, J. Perktold, in *9th Python in Science Conference* (2010)
41. T. Chen, C. Guestrin, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pp. 785–794
42. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., The Journal of Machine Learning Research **12**, 2825 (2011)
43. S.M. Lundberg, G.G. Erion, S.I. Lee. Consistent individualized feature attribution for tree ensembles (2019)
44. P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al., Nature methods **17**(3), 261 (2020)