Evaluation of Filter Methods for Feature Selection by Using Real Manufacturing Data

Alexander Gerling
Business Informatics
Furtwangen University of Applied Science
Furtwangen, Germany
e-mail: alexander.gerlinghs-furtwangen.de

Holger Ziekow
Business Informatics
Furtwangen University of Applied Science
Furtwangen, Germany
e-mail: holger.ziekow@hs-furtwangen.de

Ulf Schreier
Business Informatics
Furtwangen University of Applied Science
Furtwangen, Germany
e-mail: ulf.schreier@hs-furtwangen.de

Christian Seiffer
Business Informatics
Furtwangen University of Applied Science
Furtwangen, Germany
e-mail: christian.seiffer@hs-furtwangen.de

Andreas Hess
Business Informatics
Furtwangen University of Applied Science
Furtwangen, Germany
e-mail: andreas.hess@hs-furtwangen.de

Djaffar Ould Abdeslam
IRIMAS Laboratory
Université de Haute-Alsace
Mulhouse, France
e-mail: djaffar.ould-abdeslam@uha.fr

Abstract— **The importance of Machine Learning (ML) in the domain of manufacturing has been increasing in recent years. Especially, ML techniques are used to predict and explain errors in the production. One challenge of using ML in this domain is to deal with the often-high number of features in the datasets. However, a product defect can in many cases be traced back to a few relevant characteristics. In this paper, we investigate methods for finding reduced feature sets in the context of manufacturing. Here, the feature reduction promises two key advantages. One improvement is the prediction quality of the ML model. The second advantage concerns the explainability of a product error. With a reduction of features from the original dataset, we also reduce the search space for the product error origin. We investigate three different filter methods for feature selection based on 25 real manufacturing datasets, which are highly unbalanced. We describe the implementation of these and test them in three experimental approaches. Furthermore, we optimize the feature selection using a cost-based metric. Optimizing on the basis of the cost-based metric is shown to be in several cases more useful for reducing the number of features than well-established and frequently used classification metrics. In various experiments, we were able to improve the result and simultaneously reduce the number of features with our cost-based metric.**

Keywords: Filter-Based Feature Selection methods; Machine Learning; Cost based Metric; Production; Production data.

I. INTRODUCTION

Machine Learning (ML) has been increasingly used in the domain of manufacturing, particularly in the production line to predict corrupted product parts [1][2]. Data scientists and quality engineers are user roles in a company, which analyze malfunctions in the production to eliminate production errors. The task of these user roles is it to find the cause of a production error. Highly advanced production lines result in only few, but costly, errors. This is reflected in the data by just few errors to analyze, which result as a main challenge for this domain. A ML system can support data scientists and quality engineers, e.g., by creating prediction models and identifying relevant features in datasets from test stations. The evaluation methods for feature selection methods can be divided into five groups: embedded, hybrid, ensemble, wrapper and filter. In this paper we investigate the effect of filter selection methods. Filter methods have the advantage of better time performance in comparison to wrapper methods and are classifier independent [3]. Independence of classifier is particularly important for the flexibility to choose a different classifier regarding black box optimization. Another advantage of the filter-based methods is the ability to scale up to high-dimensional datasets [4]. This is particularly useful because a large number of measurements are recorded in a production line. Even a single test station in a production line can check numerous features, but not every feature is equally important for a classification. Unnecessary features in the dataset are features, which are not related to a specific error message. These features should be removed from the dataset to provide (1) a better result and (2) to support the explainability of resulting models. Such tasks could be solved automatically by using Automated Machine Learning (AutoML) tools. Different AutoML solutions emerged over the past few years

 ISBN: 978-1-61208-891-4

[5][6]. An AutoML tool can take over various steps of the data science pipeline and prepare data or ML models for users. Our work is part of the project PREFERML [7] that investigates challenges and holistic system solutions in this context. Within our project, we provide the user an optimized dataset including only selected features from the original dataset. The selection of features will help to reduce the error cause space and support targeted analyses. We will test three different forms of state-of-the-art filter methods on real manufacturing data.

The following research questions (Q) emerge from the above description:

- (Q1) Can state of the art filter methods provide a benefit in the given use case?
- (Q2) Which is the best filter method in our use case?
- (Q3) Can further feature reductions be achieved by using alternatives to standard metrics?
- (Q4) How long does the optimization of the model take and is there a fastest filter method?

The paper is organized as follows: Section 2 describes our use case and the challenges in this domain. In Section 3, we describe related work. The fundamentals for our experiments are described in Section 4; this includes the used metrics and the filter methods. In Section 5, we describe the filter selection algorithm. Our setup for the experiment and the different experiment approaches are described in Section 6. Section 7 is dedicated for the evaluation of the results. Section 8 summarizes the work of this paper.

II. USE CASE

In this section, we provide basic information about the manufacturing domain and describe our use case. Production lines can be equipped with different numbers of test stations. In a production line, many test stations can be arranged in sequence or in parallel. Various production errors can be detected at different test stations. While an error is detected at a specific station, measurements from preceding stations may contain clues about the error and thus it is possible to stop the production of corrupted parts earlier in the production process avoiding additional costs. To analyze this, it is important to link data from several test stations and trace back the individual products through the production line. A general description of a production line can be found in [8]. Within this scenario, we can use feature selection based on specific product errors to investigate their causes. The objective is to use machine learning on data from the test stations to predict and prevent production errors. However, the number of recorded test measurements is very high and only a fraction of the data is useful to explain specific errors. Surplus data can negatively impact the model performance. Therefore, it is important to reduce the number of features to train the ML model. Another reason to reduce the number of features is the explainability of correlations between errors and their underlying causes. After reduction, only a few features are left from the origin dataset. Thus, we also have reduced the search space for quality engineers that seek to understand error causes. A quality engineer often just has simple tools to investigate the data from the test stations. Without adjusted analyzing tools for this task, a quality engineer must search in a wide range of features for correlation with the product error. After the reduction of the dataset with a feature selection method, a quality engineer can analyze the product error with better focus on the relevant data and easier find the root cause. Another important point to be considered for ML training in our use case are the highly unbalanced datasets. These can severely impact the performance of a ML model, if not accounted for. This could be solved by using various sampling methods. Another method to tackle this challenge is to use weight parameters in learning models. In our experiments, we use weights and hyperparameter tuning to counteract the unbalanced dataset by adjusting the associated parameter with different ratios.

III. RELATED WORK

Zhang et al. [9] introduced a case study to optimize the process of a production by using feature selection methods was conducted. To do so, the authors used feature selection methods based on acceptance testing strategies. As a result, they show a reduction of 81% of inspection time while keeping the same accuracy with current industrial strategies to distinguish a non-qualified from a qualified product. An industrial strategy is e.g., acceptance sampling, which is commonly used as a statistical quality control method. The objective of the case study was to reduce the total testing time and optimize the production capability while still secure the accuracy of quality inspection for industrial products. Therefore, the reduction is not to meant to find an error cause, as in our use case. What is not considered in this reduction is that tests could be removed that would lead to the origin of a product error.

Liu et al. [10] show the problem of feature selection methods is investigated. They address the problem that standard feature selection methods do not take into account the imbalance of classes. During the selection process, the majority class is taken into account to a greater extent, which may lead to incorrectly selected features. To handle the problem, the F-measure metric was used for optimization, as it performs better on unbalanced data than accuracy does. For the investigation of the cost-sensitive classification, they generated and assigned various costs for the different classes based on a rigorous theory guidance. As result, they could reduce the number of features by optimizing with the F-measure metric. This work is similar to ours in terms of unbalanced data. [10] aim to reduce features in a cost-optimized way. Unlike the discussed work, we use a cost-based metric to optimize the results of real-world data and use further filter selection methods for our experiments.

The subject of cost-sensitive classifier and MetaCost is covered in [11]. Their approach uses a cost matrix with different costs for the errors. Afterwards, the classifier is adjusted based on this matrix. Due to the allocated costs of different errors, this approach is well suited for imbalanced datasets. A cost reduction can be accomplished with this approach compared to the cost-blind classifier. In this cost matrix the minority class was set to 0 and the majority class set to 1, based on a two-class (fail, no fail) classification. The

 ISBN: 978-1-61208-891-4

error costs for the cost matrix was set to C(0,0) = C(1,1) = 0; C(0,1) = 1000; C(1,0) = 1000 *r*. In our approach the C(0,0) and C(0,1) cases are relevant, because they correspond to a corrupted product, that is predicted as such (factor C(0,0)) and a good product, that is predicted as bad product (factor C(0,1)).

Huang et al. [12] investigate the correlation and significance among labels for multi-label data. To handle this problem, they introduce label significance into cost-sensitive feature selection. Furthermore, they suggest a feature selection algorithm, which utilize test cost based on label significance. Three distributions (namely Uniform, Normal and Pareto distribution) with positive region generate a test cost matrix, which are combined with the suggest algorithm. Moreover, by analyzing the feature cost integrated in the positive region, they define a feature significance metric. As result, they could validate the efficiency of the algorithm with the influence of an additional parameter on various test costs in their experiments and analysis of the suggested method on four real datasets.

The subject of feature selection is a popular field in different applications or domains [13][14]. One of the important reasons to use feature selection is the reduction of high dimensional data. Another reason is to select only important features to explain a certain behavior or correlation. Also, in the domain of manufacturing feature selection is also an important aid [15][16]. Our work contributes to the case of manufacturing. We are using feature selection to reduce the original dataset, which helps us to identify the origin of error causes. We are doing so by optimizing with a cost-based metric. Further, we are using filter selection methods for our experiment and use case. By using filter selection methods, we can exchange the underlying algorithm without hesitation. This characteristic helps us in terms of AutoML. Regarding this, the related works do not provide specific insights in what optimization methods and metrics work best in the manufacturing domain.

Wrapper methods for feature selection evaluate a subset of all features using a specific machine learning algorithm. These have a pre-defined search strategy to check for the best possible result from the feature subsets [26]. Wrapper methods have a high computation time, especially for datasets with many features because it must search for the best subset of features. Our advantage compared to wrapper methods is that we use the filter methods to pre-sort the most important features. Therefore, we can always replace the learning algorithm (XGBoost) in the background with another one. Furthermore, we would also have a time advantage if, for example, only the n most important features should be taken. In addition, the ordered feature list can be used for further analyses. We also go through several subsets of the features, but these are already sorted by the feature importance.

Our work is inspired by the existing works and tests several approaches for handling feature selection and hyper parameter tuning with real world data. We thereby provide insights into the benefits of different optimization metrics and strategies under realistic conditions.

IV. FUNDAMENTALS

In this section, we describe fundamental concepts behind selection methods and metrics. We first introduce three different filter methods used in our experiments and afterwards two metrics.

ANalysis Of VAriance (ANOVA) is a statistical and state of the art approach to select features in datasets. ANOVA tests the statistical significance of mean differences among different groups of scores [17]. We chose SelectKBest [18] from scikit-learn as tool to implement ANOVA filter method for our experiments. The underlying feature scores are assigned by ANOVA F-Value [19], a metric which calculates linear dependencies between two variables. The advantage of ANOVA is that if there is little or no statistical significance, these features are considered late in the ordering and can often be excluded. A disadvantage of ANOVA is that it considers only one independent feature in relation to the prediction outcome.

Kendall's rank coefficient or also called Kendall's tau (τ) is a measure for the correlation between an observation of at least two ordinally scaled features x and y. The rank correlation shows the correlation between these variables, in which no hypothesis about the statistical distribution of the variables is made [20]. An advantage of Kendall's tau is the robustness against outliers. The disadvantage of this method is that some information of the original data can be lost, for example the true distribution function. We implemented the kendalltau function from the scipy.stats package for our experiments [21].

The permutation feature importance [22] is another method to select features from a dataset. The permutation feature importance for a classifier measures the impact of a feature on the performance of a model (e.g., the accuracy). In this procedure, the performance is measured with and without permuted values of the feature. The difference between the performance with and without permuted values is computed for each model and averaged to get the feature importance see e.g., [23]. A clear advantage of this is that it can handle different metrics. This leaves us a free space for our own metrics to use. A disadvantage of permutation feature importance is the higher computational cost, compared to ANOVA or Kendall's Rank. To calculate the permutation feature importance, we must execute first an independent algorithm.

To understand the metrics, we want to clarify the groups of the confusion matrix in the context of our use case. In our experiments, we focus on the minority class because product errors occur far less than good products in a production line. This fact should be considered when choosing the metrics. Firstly, a True Positive (TP) instance is a corrupted product, that is predicted as such. A False Positive (FP) instance is a good product, that is predicted as bad product. The next group is the False Negative (FN) instance, which represent a corrupted product, that is predicted as good product. The last group is the True Negative (TN) instance. This is a good product, which is predicted as such. The explained groups of the confusion matrix are used by the upcoming metrics, to calculate the performance of the ML model.

 ISBN: 978-1-61208-891-4

The first metric we want to describe is the Matthews Correlation Coefficient (MCC) [24]. The value of the MCC metric gets calculated by:

$$\mathrm{MCC} = \frac{\mathrm{TP} * \mathrm{TN} - \mathrm{FP} * \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(TN+FN)}} \quad (1)$$

The MCC metric is a good solution for unbalanced datasets in terms of a binary classification. The MCC metric takes both classes into account and can therefore provide a good statement about the performance of the ML model. Moreover, the MCC metric has been increasingly used in the past years and is a state-of-the-art metric for a classification.

The following metric is our own cost-based metric, which is based on the cost formula of [11]. This metric is a calculation to predict the cost savings for all predictions. Symbol α represents the cost saving factor or ratio of savings of an identified error in relation to the costs of an instance incorrectly classified as an error. A quality engineer can adjust α for different products and can therefore decide product by product which ML configuration is the most suitable. Therefore, we want to use the Expected Benefit Rate (EBR) metric to maximise the possible savings for a company.

$$\text{Expected Benefit Rate} = \frac{\mathrm{TP} * \alpha - \mathrm{FP}}{\mathrm{TP}+\mathrm{FP}+\mathrm{TN}+\mathrm{FN}} \quad (2)$$

Within the EBR metric, the numerator represents the absolute savings or our expected benefit. We derive the formula from [11] and obtain the total costs by TP*C(0,0) + FP*C(0,1) + FN* C(1,0) + TN*C(1,1). Here the individual parts represent C(0,0) = α, C(0,1) = -1, C(1,0) = 0, C(1,1) = 0, which leads us to the numerator TP* α - FP. The denominator is used to normalize the savings by dividing the numerator by all instances. As result, we can show the expected benefit rate for all predictions. The EBR metric intends to demonstrate the benefits to use a ML system in production. Without a ML system, it is difficult to get hints on possible correlations in the data for the quality engineer. Without these hints on the origin of the error, we cannot save any costs, i.e., all instances must be classified as negative. Therefore, all error instances will belong to class FN. To describe a positive result, we use the terms of cost savings or benefit. Conversely, costs are negative savings or a negative benefit. By using a ML system, a previous FN could be turned into a TP, which increases savings in the production. To be precise, we would correctly predict a corrupted part that would otherwise proceed further in the production line. A FP would still produce costs but less than the savings of a TP, which we consider by α. By dividing the numerator by all predictions within the metric calculation, the result could be in the hundredths or thousandths range. The outcome result looks maybe like a small saving, but it is very profitable in mass production.

V. DESIGN OF FILTER SELECTION ALGORITHMS

This section is dedicated to the explanation of the implementation summarized as condensed pseudocode in Listing 1.

Listing 1: Pseudocode for feature selection

```
1.  Fselect(F, m, ≥r, p, T, V)
2.      S ← F, opt ← −∞
3.      Sort(F, ≥r)
4.      For i = 1 to |F|
5.          C ← {fk ∈ F | k ≤ i}
6.          score ← m(C,p,T,V)
7.          If score > opt and lp < α
8.              opt ← score
9.              S ← C
10.     Return (S)
```

The pseudocode describes the core approach for all experiments. We subsequently discuss the pseudo code and the variations for the different experiment. Line 1 defines the parameters for the feature selection. In this representation, *F* is a set of features and *m(C,p,T,V)* is a metric that evaluates a prediction mechanism *p* that is trained over data *T* and evaluated on validation data *V*. The symbol ≥*r* is an ordering relation over features according to some importance measure, with (f1, f2) ∈ ≥r if f1 is more important than f2.

At the start, we define list *S* with F (all features), in the case that there will be no better result and a variable *opt* that we define as negative infinity in Line 2. A for-loop to iterate over the number of the features |*F*| is defined in Line 4. This is to implement a version of a sequential feature selection filter method. In Line 5 we select a current subset of features *C* within the for-loop, with the features f_k from the passed ordering relation over the features ≥*r*. For every iteration, the next feature from F ordered by ≥*r* is added to *C*. In Line 6 we calculate the *score* based on the passed metric *m(C,p,T,V)* to optimize. Note, that m wraps the training process for the predictor p and is an important design choice. One may make compromises concerning the implementation for the sake of reducing computation time, e.g., implement m with or without hyperparameter optimization.

The calculation of *score* is followed by a check if the calculated *score* is greater than the *opt* variable and that *lp* is smaller than α. The *lp* variable represents the p-value from the two-sided dependent T-test [25] and α is set to 0.05. The p-value is calculated for the baseline model, which represent the model with all features trained and the current model within an iteration of the for-loop, to make sure that our current model is significantly better than the original model. If this condition is met, we update *opt* with the *score* value *in Line 8* and set *C* as *S in Line 9*. At the end, we return the feature list *S* in Line 10.

VI. EXPERIMENTS

We ran several experiments based on the approach that we introduced in the previous section. These experiments differ (1) in the way hyperparameter tuning is integrated and (2) in the implementations for the ordering relation $\geq_r$.

We integrate hyperparameter tuning in three different ways. The basic approach (A) does not include any hyperparameter tuning. The predictor with default parameters is trained and used on data projected on the selected features. Experiment approach (B) adds hyperparameter tuning as subsequent step to approach (A). After returning *S* at the end of Listing 1, we reduce the original dataset to the selected features and optimize a new model with the parameters from TABLE 3. For the optimization via hyperparameter tuning, we went through all possible combinations of the parameters from TABLE 3, i.e., we implemented a grid search strategy. With this new optimized model, we create the final results. The third experiment approach (C) integrates hyperparameter tuning into the basic experiment (A) by optimizing the parameters of every single model during the training of mechanism m in line 5. At the end of the function, we return the best result and the selected features.

We consider three alternative methods in order to create the ordering $\geq_r$. As a first case (a), features are ordered according to measurements based on ANOVA. An alternative sorting (b) is provided by Kendall's rank coefficient. As third method (c) we chose permutation feature importance. This procedure requires an additional ML model in advance to calculate the importance of each feature. Based on this, we create the ordered feature list using a certain metric, which was EBR or MCC, depending on the optimization.

We ran experiments with all possible settings of hyperparameter tuning. These include three variations with respect to the feature ranking $\geq_r$, yielding a number of 18 sets of experiments in total (Optimizing according to EBR and MCC). To optimize the model, we used the training set for training and test set to evaluate the model. We continue to perform a final 10-fold cross validation with a T-test on the optimization metric based on the best model. Therefore, we ensure that the final model is not worse than the baseline model based on the training set. If so, we use the baseline model instead of the optimized model.

A. Test Setup

For our experiments, we used a machine with Windows 10 64Bit. The test system has an Intel(R) Xeon(R) W-2133(12x 3.60 GHz) processor and 32 GB RAM. We used the Anaconda Distribution with Numpy Version: 1.18.1, Pandas Version: 1.0.1, Scikit-learn Version: 0.22.1 and Python 3.7.6. All shown experiments were executed on the CPU. For the experiments, we used the well-known XGBoost algorithm with the version 0.90. XGBoost is a state-of-the-art algorithm to predict product quality [27]. Furthermore, the comprehensibility of the results is an important criterion for quality engineers, which is most likely to be fulfilled by decision trees [8].

TABLE 3: XGBOOST OPTIMIZATION PARAMETER

Hyperparameter	
n_estimators	50, 100, 150
max_depth	3, 6, 9
learning_rate	0.1, 0.3
class_weight	({0:1, 1:1}, ({0:1, 1:10}), ({0:1, 1:int(M)}), (M = (sum(negative instances) / sum(positive instances))

All optimizations for the experiment approaches B and C have been calculated using the parameter search space from TABLE 3.

B. Data Preparation for Training

To prepare the datasets for classification, we used a sequential split. The data is ordered by time. We set the split for the training set to first 67% of the total amount of errors in the data. Therefore, we have always 33% of the total amount of errors in the test set to validate the quality of the ML model.

C. Datasets

For our experiments, we used 25 highly unbalanced datasets from six different production lines. Within these production lines, we took the measurements from various sequential test stations and addressed various error messages. TABLE 2 shows the ratio between good and corrupted product parts.

TABLE 2: DATASETS

Dataset	Class 0	Class 1	Instances	Features	IR (Class 1 / Class 0)
A	57499	530	58029	64	0.009218
B	7127	73	7200	17	0.010243
C	88928	553	89481	23	0.006219
D	67065	1885	68950	133	0.028107
E	55204	1570	56776	29	0.028440
F	42894	1187	44081	33	0.027673
G	59321	245	59566	30	0.004130
H	43373	182	43555	90	0.004196
I	58345	799	59144	103	0.013694
J	55473	139	55612	64	0.002506
K	58585	1747	60332	19	0.029820
L	194318	614	194932	22	0.003160
M	6867	33	6900	34	0.004806
N	86664	388	87052	99	0.004477
O	6939	129	7068	38	0.018591
P	189809	233	190042	53	0.001228
Q	87292	388	87680	34	0.004445
R	43356	199	43555	90	0.004590
S	6854	33	6887	96	0.004815
T	11228	123	11351	22	0.010955
U	11349	48	11397	54	0.004229
V	13029	212	13241	30	0.016271
W	10604	117	10721	102	0.011034
X	89204	687	89891	17	0.007701
Y	190183	400	190583	20	0.002103

Class 0 represents a good and Class 1 represents a corrupted product part. The imbalance of the two classes is shown by the column IR. Dataset K has the highest IR value

with 0.02982 and dataset P the lowest with 0.001228. This fact stresses the importance of considering imbalance in the domain of manufacturing, in particular for highly optimized production lines. For our experiments we use numerical and categorical data. Numerical values are measurements from one or several combined test stations.

A further point is the number of features of the datasets. Especially interesting are the datasets D, H, I, N, R, S, W because of the high number of features (90+). The effect of the feature reduction should be seen clearly on these.

VII. EVALUATION

In this section, we present the results from the executed experiments. We used all three presented feature ranking methods for each optimization approach. The objective was to find out, which combination of ranking and optimization approach is the most suitable regarding the prediction quality and execution time. To evaluate the results with one key figure we used the EBR metric. Even if we optimize according to MCC we calculate the EBR value to compare. We set α=10 in our experiments. According to our project partner, this is a reasonable assumption for α in many cases. However, the specific values may vary greatly throughout the production lines and error types. Yet, we keep a fixed number to make the results on different experiments comparable. For the analysis, we first compared the EBR value and the number of necessary features.

InTABLE 3, we show the baseline results of the test without any optimization, filter methods or the use of the Fselect function. These results are our baseline to compare later results and were created with the standard settings from the XGBoost algorithm. For the baseline results, we do not consider the imbalance of the classes. There are several aspects to point out inTABLE 3. First, some datasets have a high number of features. The next point is the EBR value. If an EBR value is 0, this does not mean that the model did not find a relation in the data, but the predicted error probabilities are too low for making an economically reasonable error prediction (i.e., the cost for false positives would outweigh the savings through true positives and hence TP = 0, FP = 0). These results do not provide a meaningful prediction, but they show a possible hint to the quality engineer regarding the error causes. A further point is the negative EBR value. In this case, the model estimated the confidence in error prediction too high, resulting in higher cost through false positive predictions than savings through true positive results.

TABLE 3: BASELINE RESULTS
(MODELL TRAINED WITH ALL FEATURES)

Dataset	Features	EBR Value	Dataset	Features	EBR Value
A	64	0.00042199	N	99	-0.000073042462
B	17	0.03246753	O	38	-0.003707627
C	23	0.00050081	P	53	0
D	133	0.21394069	Q	34	-0.000193916
E	29	0.00270392	R	90	0.074146982
F	33	0.00084728	S	96	0
G	30	0	T	22	0
H	90	-0.00040975	U	54	0
I	103	0.00157487	V	30	0.001569859
J	64	0	W	102	0.002081165
K	19	0	X	17	0.001777727
L	22	0	Y	20	0
M	34	-0.00021906			

TABLE 4 shows the results of the experiment approach A. To highlight the best results (best EBR, using the number of features as tie-breaker) for a dataset in TABLE 4 and following tables, these lines are colored with a green background color. If there is no green background color in a

TABLE 4: EXPERIMENT APPROACH A

Dataset Name	ANOVA				Kendall's rank coefficient				Permutation Feature Importance			
	Optimized according to EBR		Optimized according to MCC		Optimized according to EBR		Optimized according to MCC		Optimized according to EBR		Optimized according to MCC	
	Features	EBR Value	Features	EBR Value	Features	EBR Value	Features	EBR Value	Features	EBR Value	Features	EBR Value
A	64	0.000421987	64	0.000421987	64	0.000421987	64	0.000421987	64	0.000421987	64	0.000421987
B	17	0.032467532	17	0.032467532	17	0.032467532	17	0.032467532	17	0.032467533	17	0.032467533
C	23	0.000500814	17	0.00109553	23	0.000500814	23	0.000500814	23	0.000500814	23	0.000500814
D	2	0.213996855	2	0.213996855	1	0.213996855	1	0.213996855	133	0.213940688	133	0.213940688
E	2	0.000600871	2	0.000600871	7	0.000600871	7	0.000600871	2	0.00135196	3	0.00135196
F	33	0.000847278	33	0.000847278	33	0.000847278	33	0.000847278	33	0.000847278	4	0.00021182
G	1	0	30	0	1	0	30	0	1	0	30	0
H	1	0	1	0	1	0	1	0	1	0	1	0
I	1	0	89	0.00157487	1	0	103	0.00157487	1	0.001532306	1	0.001532306
J	64	0	64	0	64	0	64	0	64	0	64	0
K	1	0	1	0	1	0	1	0	1	0	1	0
L	22	0	22	0	22	0	22	0	22	0	22	0
M	34	-0.000219058	34	-0.000219058	34	-0.000219058	34	-0.000219058	34	-0.000219058	34	-0.000219058
N	1	0	2	-0.032576938	1	0	1	0	99	-0.000073042462	99	-0.000073042462
O	38	-0.003707627	38	-0.003707627	38	-0.003707627	38	-0.003707627	38	-0.003707627	38	-0.003707627
P	53	0	53	0	53	0	53	0	53	0	53	0
Q	1	0	2	-0.034856381	1	0	1	0	34	-0.000193916	34	-0.000193916
R	4	0.067585302	9	0.067585302	4	0.067585302	1	0.077427822	5	0.077755906	5	0.077755906
S	96	0	31	0	74	0	25	0	16	0	18	0
T	22	0	5	0	22	0	22	0	22	0	16	0
U	54	0	54	0	54	0	54	0	54	0	54	0
V	30	0.001569859	30	0.001569859	30	0.001569859	2	0.001098901	30	0.001569859	30	0.001569859
W	102	0.002081165	102	0.002081165	102	0.002081165	102	0.002081165	102	0.002081166	6	0.004682622
X	3	0.002599687	2	0.002007111	3	0.002599687	2	0.002007111	1	0.001835073	1	0.001835073
Y	20	0	20	0	20	0	20	0	20	0	20	0

 ISBN: 978-1-61208-891-4

TABLE 5: EXPERIMENT APPROACH B

Dataset Name	ANOVA				Kendall's rank coefficient				Permutation Feature Importance			
	Optimized according to EBR		Optimized according to MCC		Optimized according to EBR		Optimized according to MCC		Optimized according to EBR		Optimized according to MCC	
	Features	EBR Value	Features	EBR Value	Features	EBR Value	Features	EBR Value	Features	EBR Value	Features	EBR Value
A	64	-0.014659475	64	-0.000990753	64	-0.014659475	64	-0.000990753	64	-0.014659475	64	-0.000990753
B	17	0.032467532	17	0.032467532	17	0.032467532	17	0.032467532	17	0.032467533	17	0.032467533
C	23	0.000500814	17	0.019531739	23	0.000500814	23	0.019531739	23	0.000500814	23	0.019531739
D	2	0.213996855	2	0.213996855	1	0.213996855	1	0.213996855	133	0.213940688	133	0.213940688
E	2	0.029968454	2	0.000600871	7	0.031320415	7	0.004206099	2	0.000225327	3	0.00135196
F	33	0.000847278	33	0.008825814	33	0.000847278	33	0.008825814	33	0.000847278	4	0.018428299
G	1	0	30	0	1	0	30	0	1	0	30	0
H	1	0	1	-0.233302602	1	0	1	0	1	0	1	-0.233302602
I	1	0	89	0.000595897	1	0	103	0.00157487	1	0.001532306	1	-0.632459351
J	64	0	64	-0.3219757	64	0	64	-0.3219757	64	0	64	-0.3219757
K	1	0	1	-0.21820103	1	0	1	-0.356447849	1	0	1	-0.119182847
L	22	0	22	0	22	0	22	0	22	0	22	0
M	34	-0.000219058	34	-0.000219058	34	-0.000219058	34	-0.000219058	34	-0.000219058	34	-0.000219058
N	1	-0.001436502	2	-0.651952668	1	0	1	-0.965450915	99	-0.151806584	99	-0.151806584
O	38	-0.003707627	38	-0.003707627	38	-0.003707627	38	-0.003707627	38	-0.003707627	38	-0.003707627
P	53	0	53	0	53	0	53	0	53	0	53	0
Q	1	0	2	-0.965604169	1	0	1	-0.163955884	34	-0.000193916	34	-0.05189674
R	4	0.125	9	0.074146982	4	0.125	1	0.163385827	5	0.130249344	5	0.077755906
S	96	0	31	0	74	0	25	0	16	0	18	0
T	22	0	5	0	22	0	22	0	22	0	16	0
U	54	0	54	0	54	0	54	0	54	0	54	0
V	30	0.001569859	30	0.006279435	30	0.001569859	2	-0.001098901	30	0.001569859	30	0.006279435
W	102	0.012747138	102	0.004162331	102	0.012747138	102	0.004162331	102	0.012747138	6	0.002341311
X	3	0.01062813	2	0.010226708	3	0.01062813	2	0.010226708	1	0.009137134	1	0.009997324
Y	20	0	20	-0.066429903	20	0	20	-0.066429903	20	0	20	-0.066429903

line, there are only identical results and therefore no winner. For this experiment we also used the standard parameters for the algorithm and did not consider the unbalanced dataset. Compared to Table 3 we improved the result in 16 out of 75 experiments based on the EBR optimization. However, this is contrasted by eight deteriorations compared to the baseline. One of the possible reasons for the deterioration of results is a concept drift in the data. During the training, a model was found that performed better on the training set but afterward worse based on the test set. This is because the production processes are subject to constant change. With these results, we can show that our safety mechanism based on the T-test is working. That is, we avoid significant deterioration through failed optimization while gaining benefits when the optimization works. We can already notice an improvement in approach A compared to the baseline results. We were able to reduce dataset X to 3 out of 17 features with ANOVA. Dataset D could be reduced from 133 to 1 feature using

TABLE 6: EXPERIMENT APPROACH C

Dataset Name	ANOVA				Kendall's rank coefficient				Permutation Feature Importance			
	Optimized according to EBR		Optimized according to MCC		Optimized according to EBR		Optimized according to MCC		Optimized according to EBR		Optimized according to MCC	
	Features	EBR Value	Features	EBR Value	Features	EBR Value	Features	EBR Value	Features	EBR Value	Features	EBR Value
A	30	-0.124339498	64	0.000421987	44	-0.095901218	47	-0.056032585	19	-0.005082196	29	0.002843828
B	2	0.037962038	17	0.032467532	3	0.061938062	17	0.032467532	2	0.058941059	14	0.044955045
C	5	0.019093527	23	0.000500814	5	0.019093527	5	0.019250031	1	0.013334168	1	0.0134593715
D	2	0.213996855	133	0.213940687	2	0.213996855	2	0.213996855	133	0.213940688	133	0.213940688
E	3	0.027264534	29	0.002703921	7	0.031320415	7	0.002103049	5	0.031921286	6	0.003530119
F	25	0.003177293	33	0.000847278	30	0.005154275	30	0.005154275	14	0.014333122	12	0.014756761
G	1	0	30	0	1	0	16	-0.045876679	1	0	2	-0.069512535
H	1	0	90	-0.000409752	1	0	3	-0.214146691	1	0	29	-0.020692481
I	1	0	103	0.00157487	23	0.001234358	23	0.002085639	1	0.001532306	1	-0.632459351
J	64	0	64	0	64	0	24	-0.102042613	64	0	3	-0.387920409
K	1	0	19	0	1	0	6	-0.226214062	1	0	1	-0.119182847
L	22	0	22	0	22	0	22	0	22	0	2	-0.367024358
M	34	-0.000219058	34	-0.000219058	34	-0.000219058	34	-0.000219058	34	-0.000219058	2	-0.120920044
N	1	-0.001436502	99	-0.000073042462	1	0	1	-0.965450915	61	-0.239652318	10	-0.350311648
O	4	-0.094632768	38	-0.003707627	1	-0.01059322	20	-0.000706215	9	-0.002295198	8	-0.000176554
P	53	0	53	0	53	0	53	0	53	0	53	0
Q	2	-0.128663192	34	-0.000193916	1	0	1	-0.163955884	34	-0.000193916	34	-0.05189674
R	2	0.107611549	90	0.074146982	2	0.107611549	4	0.125	2	0.107611549	9	0.144685039
S	40	0.002192982	96	0	29	0	6	-0.000877193	14	0	14	0
T	5	0	22	0	22	0	5	0	22	0	1	-0.235951417
U	54	0	54	0	54	0	10	0.000895255	54	0	12	-0.002088929
V	2	-0.00266876	30	0.001569859	3	-0.000784929	2	-0.000941915	1	0.003296703	30	0.006279435
W	29	0.010665973	102	0.002081165	8	0.014308012	6	0.008584807	6	0.020031218	15	0.00364204
X	3	0.01062813	17	0.001777727	3	0.01062813	3	0.01062813	1	0.009137134	1	0.009997324
Y	20	0	20	0	20	0	7	-0.044448366	20	0	6	-0.075010754

 ISBN: 978-1-61208-891-4

TABLE 7: EXECUTION TIME COMPARISON

Dataset Name	Experiment Approach A			Experiment Approach B			Experiment Approach C		
	ANOVA Execution Time	Kendall's rank coefficient Execution Time	Permutation Feature Importance Execution Time	ANOVA Execution Time	Kendall's rank coefficient Execution Time	Permutation Feature Importance Execution Time	ANOVA Execution Time	Kendall's rank coefficient Execution Time	Permutation Feature Importance Execution Time
A	0:00:48	0:00:48	0:00:47	0:02:54	0:02:54	0:02:55	1:25:45	1:33:30	1:26:19
B	0:00:09	0:00:09	0:00:07	0:00:51	0:00:51	0:00:49	0:09:33	0:10:44	0:09:36
C	0:01:17	0:01:16	0:01:16	0:09:36	0:09:18	0:09:16	2:01:46	2:13:55	2:01:46
D	0:21:54	0:22:51	0:25:27	0:23:27	0:23:37	1:03:34	41:04:49	47:06:54	47:58:57
E	0:01:16	0:01:21	0:01:11	0:02:51	0:04:24	0:02:36	2:24:09	2:40:13	2:14:30
F	0:01:12	0:01:11	0:01:05	0:07:19	0:07:09	0:06:59	2:10:01	2:24:19	1:59:42
G	0:01:04	0:01:00	0:01:04	0:02:10	0:02:03	0:02:11	2:01:41	2:05:21	2:01:08
H	0:04:36	0:04:35	0:04:45	0:05:24	0:05:04	0:05:40	8:35:13	9:13:34	8:53:49
I	0:17:07	0:15:38	0:15:30	0:18:19	0:16:36	0:16:38	32:33:45	33:00:00	29:47:19
J	0:05:26	0:05:21	0:05:25	0:20:25	0:20:25	0:20:24	9:20:57	9:54:35	9:17:51
K	0:00:37	0:00:38	0:00:39	0:01:25	0:01:14	0:01:49	1:02:49	1:08:30	1:11:56
L	0:03:05	0:03:05	0:02:58	0:25:21	0:25:18	0:25:08	5:04:10	5:33:01	4:50:34
M	0:00:08	0:00:09	0:00:09	0:00:41	0:00:42	0:00:42	0:14:31	0:16:18	0:14:33
N	0:10:26	0:10:19	0:10:55	0:11:45	0:11:04	0:31:19	17:56:08	19:27:29	18:37:31
O	0:00:08	0:00:09	0:00:08	0:00:43	0:00:43	0:00:43	0:17:39	0:19:18	0:17:51
P	0:12:02	0:12:03	0:12:07	0:53:41	0:53:13	0:53:56	21:01:48	22:33:36	21:19:08
Q	0:01:40	0:01:39	0:01:51	0:03:03	0:02:26	0:10:06	2:51:35	3:04:39	3:00:01
R	0:07:39	0:07:35	0:07:19	0:09:11	0:09:03	0:09:11	10:48:54	11:48:45	10:27:05
S	0:00:45	0:00:44	0:00:47	0:01:46	0:01:38	0:01:13	1:04:45	1:11:51	1:06:39
T	0:00:09	0:00:09	0:00:10	0:01:13	0:01:13	0:01:13	0:16:26	0:18:17	0:16:57
U	0:00:43	0:00:44	0:00:44	0:02:32	0:02:34	0:02:34	1:06:24	1:12:42	1:07:40
V	0:00:19	0:00:19	0:00:20	0:01:55	0:01:55	0:01:55	0:35:19	0:38:05	0:35:08
W	0:02:07	0:02:03	0:02:05	0:05:20	0:05:16	0:05:21	3:28:33	3:41:16	3:27:00
X	0:00:31	0:00:32	0:00:31	0:02:13	0:02:13	0:01:41	0:55:23	1:00:46	0:55:43
Y	0:02:06	0:02:07	0:02:06	0:18:11	0:19:31	0:18:11	3:18:35	3:34:56	3:16:28

Kendall's rank. For dataset I, we reduced the number of features from 103 to 1 using Permutation Feature Importance. In TABLE 4, we show that the ANOVA selection method was the best for approach A. In TABLE 5, we show the results of experiment approach B. For Dataset X (ANOVA), parameter optimization improved the result from 0.002599687 to 0.01062813 for the same number of features. However, some results are already optimized by a reduction to the most important features e.g., dataset D with Kendall's rank or dataset I with permutation feature importance. In this approach we could provide 21 out of 75 better and nine worse results based on the EBR optimization compared to TABLE 3. Therefore, we show a benefit to adjust the parameter of the algorithm to provide better results with this approach. With approach B we improved 11 out of 75 results and six out of 75 got worse based on the EBR optimization compared to approach A. For approach B the Kendall's rank provided the best method for the feature selection if we consider the results from the EBR and MCC optimization.

In TABLE 6, we visualized the results from approach C. Within this table we can show the most changes in the number of features and the difference between the optimization metric. First, we provide 16 out of 25 best results based on the EBR and MCC optimization with the Kendall's rank selection method in this approach. We also could reduce in 21 out of 75 cases the number of features and improve the result by optimizing with the EBR metric. In contrast, we only could reduce and improve the results twice by optimizing with the MCC metric. Here, we can clearly show the benefit to use our cost-based metric. Compared to approach B we improve 20

TABLE 8: RESULT OVERVIEW

	Number of tests where BEST results is with EBR	Number of tests where BEST results is with MCC	Number of tests where BEST results is with EBR and Features reduced	Number of tests where BEST results is with MCC and Features reduced	Number of tests where optimizing with EBR is BETTER than baseline	Number of tests where optimizing with MCC is BETTER than baseline	Number of tests where optimizing with EBR is WORSE than baseline	Number of tests where optimizing with MCC is WORSE than baseline
Approach A, ANOVA	3	2	2	1	5	4	3	4
Approach A, Kendall	2	2	0	1	2	3	3	2
Approach A, Permutation	1	1	0	1	5	3	2	3
Approach B, ANOVA	10	5	3	1	7	4	3	9
Approach B, Kendall	9	5	0	1	5	7	1	7
Approach B, Permutation	8	6	0	1	5	6	4	9
Approach C, ANOVA	10	6	9	0	9	0	6	0
Approach C, Kendall	12	6	1	1	8	9	4	11
Approach C, Permutation	15	7	4	2	10	10	3	12

 ISBN: 978-1-61208-891-4

results and got worse in 15 cases based on the EBR result. As mentioned before, the deterioration of the results may be due to a concept drift in the data.

In TABLE 7, we compare the computation time needed for each experiment approach for different datasets. We colored the best time results with different colors for each approach. For the experiment approach A the permutation feature importance selection method achieved in 11 out of 25 cases the best results. At experiment approach B the Kendall's rank selection method achieved in 13 out of 25 cases the best time results. For experiment approach C, the ANOVA selection method achieved in 15 out of 25 datasets the best time result.

When comparing experiment approach A and C in terms of the EBR result and required time, we can point out that the use of hyperparameter tuning does show a significant enhancing effect in most of our experiments. However, the calculation time for 19 datasets was demanding in terms of time (over one hour needed), especially in dataset D. With the experiment approach A and the benefit of EBR metric, we can demonstrate a significant advantage towards the baseline. Nevertheless, the experiment approach C could be used to obtain the best possible result. We summarize all experiments and results in a brief overview in TABLE 8. In this table we can show that an optimization according to MCC achieves better results than the baseline, but also often worsens especially in Approach C Kendall's rank and permutation feature importance. Therefore, we recommend optimization according to EBR.

VIII. CONCLUSION

In this paper, we showed three filter methods and used adapted cost-based metric EBR, to reduce features in real manufacturing datasets. Regarding the research questions from the introduction, we demonstrate benefits in a real-world use case, which answers Q1. We showed a benefit by using different filter methods and optimizing the XGBoost algorithm with the EBR metric. However, the different filter methods overall yield similar results. We obtain most of the best results with experiment approach C. Experiment approach B is favorable with respect to computation time. These findings provide insights on Q2. Overall, most of the best results for the experiment approaches were achieved by using the permutation feature importance selection method based on TABLE 8. Moreover, we have shown that more features of the dataset can be reduced when using the EBR metric compared to the MCC metric. This answers our question Q3. The answer for question Q4 depends on the experiment approach. The time difference between experiment approaches A and B is tolerable for better results. The training duration of a model is especially important to consider as soon as many models must be trained in parallel for different products. Especially because there are only limited computing resources. However, the Kendall's rank selection method could be used in combination with experiment approach B as fastest method regarding the best possible results. To summarize our contributions in this paper, we state the following:

First, we showed benefits of feature reduction in our use case with highly unbalanced real-world data. Second, using our EBR metric reduces the number of features in comparison to the MCC in our experiments. Third, the experiment approach B indicates the best improvement compared to the baseline regarding the computation time.

ACKNOWLEDGEMENTS

This project was funded by the German Federal Ministry of Education and Research, funding line "Forschung an Fachhochschulen mit Unternehmen (FHProfUnt) ", contract number 13FH249PX6. The responsibility for the content of this publication lies with the authors. Also, we want to thank the company SICK AG for the cooperation and partial funding.

REFERENCES

[1] Z. Li, Z. Zhang, J. Shi, and D. Wu, "Prediction of surface roughness in extrusion-based additive manufacturing with machine learning," Robotics and Computer-Integrated Manufacturing, vol. 57, pp. 488-495, 2019 doi:10.1016/j.rcim.2019.01.004.

[2] V. Hirsch, P. Reimann, and B. Mitschang, "Data-Driven Fault Diagnosis in End-of-Line Testing of Complex Products," 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2019, doi:10.1109/dsaa.2019.00064.

[3] J. C. Ang, A. Mirzal, H. Haron, and H. N. A. Hamed, "Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection," IEEE/ACM transactions on computational biology and bioinformatics, vol. 13, no. 5, pp. 971-989, 2015.

[4] Y. Peng, Z. Wu, and J. Jiang, "A novel feature selection approach for biomedical data classification," Journal of Biomedical Informatics, vol. 43, no. 1, pp. 15-23, 2010.

[5] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA," Automated Machine Learning The Springer Series on Challenges in Machine Learning, pp. 81-95, 2019, doi:10.1007/978-3-030-05318-5_4.

[6] E. LeDell and S. Poirier, "H2o automl: Scalable automatic machine learning," In Proceedings of the AutoML Workshop at ICML vol. 2020, 2020.

[7] H. Ziekow et al., "Proactive Error Prevention in Manufacturing Based on an Adaptable Machine Learning Environment," From Research to Application, vol. 113, 2019.

[8] A. Gerling et al., "A Reference Process Model for Machine Learning Aided Production Quality Management," Proceedings of the 22nd International Conference on Enterprise Information Systems, 2020, doi:10.5220/0009379705150523.

[9] Y. Zhang, E. Tochev, S. Ratchev, and C. German, "Production process optimization using feature selection methods," Procedia CIRP, vol. 88, pp. 554-559, 2020.

[10] M. Liu et al., "Cost-sensitive feature selection by optimizing F-measures," IEEE Transactions on Image Processing, vol. 27m no. 3, pp. 1323-1335, 2017.

[11] P. Domingos, "MetaCost: A General Method for Making Classifiers Cost-Sensitive," Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '99, 1999, doi:10.1145/312129.312220.

[12] J. Huang, W. Qian, B. Wu, and Y. Wang, "Cost-Sensitive Feature Selection Based on Label Significance and Positive Region," In 2019 International Conference on Machine Learning and Cybernetics (ICMLC), pp. 1-7, 2019.

 ISBN: 978-1-61208-891-4

[13] M. Ali and T. Aittokallio, “Machine learning and feature selection for drug response prediction in precision oncology applications,” Biophysical reviews, vol. 11, no. 1, pp. 31-39, 2019.

[14] Y. Liu, J. W. Bi, and Z. P. Fan, “Multi-class sentiment classification: The experimental comparisons of feature selection and machine learning algorithms,” Expert Systems with Applications, vol. 80, pp. 323-339, 2017.

[15] H. Liu, M. Zhou, and Q. Liu, “An embedded feature selection method for imbalanced data classification,” IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 3, pp. 703-715, 2019.

[16] F. Feng, K. C. Li, J. Shen, Q. Zhou, and X. Yang, “Using cost-sensitive learning and feature selection algorithms to improve the performance of imbalanced classification,” IEEE Access, vol. 8, pp. 69979-69996, 2020.

[17] B. G. Tabachnick and L. S. Fidell, “Experimental designs using” ANOVA, Belmont, CA: Thomson/Brooks/Cole, pp. 724, 2007.

[18] SelectKBest, Sklearn.feature_selection.SelectKBest, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html, (n.d.). Retrieved October 01, 2020

[19] F_classif, Sklearn.feature_selection.f_classif, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html, (n.d.). Retrieved October 05, 2020.

[20] K. Siebertz and D. van Bebber, Statistische versuchsplanung. T. Hochkirchen (Ed.). Springer Berlin Heidelberg, 2010.

[21] Kendalltau, Scipy.stats.kendalltau, from https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.kendalltau.html, (n.d.). Retrieved October 05, 2020.

[22] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, “Permutation importance: a corrected feature importance measure,” Bioinformatics, vol. 26, no 10, pp. 1340-1347, 2010.

[23] G. Casalicchio, C. Molnar, and B. Bischl, “Visualizing the feature importance for black box models,” In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, Cham, pp. 655-670, September 2018.

[24] B.W. Matthews, “Comparison of the Predicted and Observed Secondary Structure of T4 Phage Lysozyme,” Biochimica Et Biophysica Acta (BBA) - Protein Structure, vol. 405, no. 2, pp. 442–51, 1975, https://doi.org/10.1016/0005-2795(75)90109-9.

[25] SciPy.org, scipy.stats.ttest_ind - SciPy v1.6.3 Reference Guide. (n.d.). https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html, May 2021.

[26] Y. B. Wah, N. Ibrahim, H. A. Hamid, S. Abdul-Rahman, and S. Fong, “Feature Selection Methods: Case of Filter and Wrapper Approaches for Maximising Classification Accuracy,” Pertanika Journal of Science & Technology, vol. 26, no. 1, 2018.

[27] N. Zhou, Q. Ren, and J. Zhou, “Identification of Critical-to-quality Characteristics Based on Improved XGBoost,” In Proceedings of the 3rd International Conference on Data Science and Information Technology, pp. 205-209, July 2020.

 ISBN: 978-1-61208-891-4