The International Journal of Advanced Manufacturing Technology
https://doi.org/10.1007/s00170-025-16491-x

ORIGINAL ARTICLE

# Semi-supervised tapping wear detection in nodular cast iron workpieces under real industrial conditions

**Jose Alberto Maestro-Prieto**[1] · **Alain Gil-Del-Val**[2,3] · **Andres Bustillo**[1]

Received: 4 July 2025 / Accepted: 31 August 2025


**Abstract**
The tapping of metal components is a manufacturing task with great potential for automation, because the conditions affecting the industrial components are of limited variability. However, automation encounters two main problems: both the human- and the time-related costs associated with the manual classification of threads are excessive, and thread quality can vary greatly, due to tapping tool wear. In this study, the use of semi-supervised algorithms is proposed to improve the performance of machine learning–based models trained on real industrial datasets. The strategy was validated on a dataset of more than 7000 threads produced with 36 different tapping tools under the same working conditions involving nodular cast iron workpieces. Several algorithms were trained using datasets with different features and data processing. The best results were obtained with datasets using linear regression in which sinusoidal fluctuations in the raw signals were replaced by linear regressions and the slope of an 11-element rolling window was applied to extend the raw dataset. Algorithms were trained with different percentages of labeled datasets. The co-training-based algorithms almost systematically obtained the best results, yielding better results than the reference algorithms using a 100% labeled dataset. Besides, the proposed solution also achieved higher performance with 50% of labeled instances in the training dataset, drastically reducing the costs of manual labeling for that sort of industrial dataset.

**Keywords** Semi-supervised learning · Fault detection · Tapping · Wear

## 1 Introduction

Opening a hole in a metallic workpiece is a type of machining process called hole-making. Some hole-making operations also require an internal thread, with which a detachable though high-strength join can fasten different parts of the same or different materials. The process is therefore found in many industrial applications, such as the manufacturing of automotive, railway, and aeronautical components. As pointed out in [29], the threading operation is typically the last or nearly the last step of processing the part, so any machining error leads either to the rejection of a value-added part nearing completion (e.g., on-board aeronautical components) or additional costly manual work to correct the failure. A risk turns a simple process, such as threading, into a critical task in many manufacturing processes.

There are three ways to create a thread in metallic components: thread milling, thread forming, and tapping [21]. In thread milling, the tool is slightly smaller in diameter than the pre-drilled hole where the thread will be cut, in a process similar to milling. Thread forming (also called turning) uses a tool to deform rather than to cut the material, so no chips are produced and better results are obtained in highly malleable materials. Tapping is similar to thread milling, but the tool diameter and the pre-drilled hole are nearly the same

Alain Gil-Del-Val and Andres Bustillo contributed equally to this work.

✉ Andres Bustillo
abustillo@ubu.es

Jose Alberto Maestro-Prieto
jamaestro@ubu.es

Alain Gil-Del-Val
alain.gildelval@unir.net; alain.gil@tecnalia.com

[1] Departamento de Ingeniería Informática, Universidad de Burgos, Avda. Cantabria, s/n, 09006 Burgos, Spain

[2] Escuela Superior de Ingeniería y Tecnología, International University of La Rioja UNIR, Avenida de la Paz, 137, 26006 Logroño, La Rioja, Spain

[3] TECNALIA, Basque Research and Technology Alliance (BRTA), Parque científico y Tecnológico de Guipuzkoa, Donostia-San Sebastián 20009, Guipuzkoa, Spain

Published online: 19 September 2025

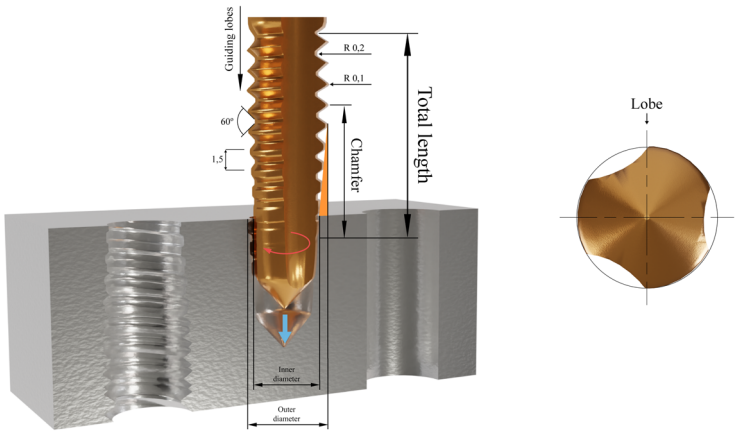


**Fig. 1** Geometry of tapping operations

diameter. Internal threading can either be done in a blind hole that does not go through a part or as a hole that goes through a part. Tapping is a very complex operation, especially for go-through taps, because different threading stages require different feed and rotation speeds, so both the rotation and feed motion of the machine must be synchronized.

Tapping is performed with a tap, a cutting tool with a unique profile (see Fig. 1). The cutting portion of the tap is tapered with a number of teeth. Its conical shape ensures that friction progressively increases as the tap perforates the workpiece. Tool geometry greatly influences the type of tool failure. According to [29], a cutting tool can fail in four ways: (*1*) tool deformation, (*2*) chipping due to mechanical breakage, (*3*) overheating, and (*4*) progressive wear. Under industrial conditions, tap breakage is very rare, because taps are changed long before the tool weakens. The most common failure mode is therefore when progressive wear affects the threading that eventually exceeds the tolerance limits of a workpiece. The wearing processes of taps have been extensively described: teeth blunten as they wear, which increases energy consumption and heat production during cutting. Heat can cause teeth to soften and wear, leading to chipping at the tip, which may result in poor quality tapped thread flanks, smaller thread diameters, and ultimately unacceptable threading [20].

Although tapping might seem to be a simple process, due to its high repeatability in industrial components under the same processing conditions, tap life depends on many parameters such as cutting speed, feed rate, depth of cut, tool geometry, cutting tool material, tool coating, and workpiece material. Other issues that may arise include chip handling and tool lubrication. Lack of synchronism and rigidity of the tap can result in uneven threads. Successful threading of certain materials requires specific synchronization and tap hardness patterns. The numerous cutting edges of the tap and the intricate coordination between its rotational and feed movements make high-speed operations particularly challenging. All these issues leave ample room for further tapping research, as tapping optimization is a high priority industrial need. Some authors, such as [29], have proposed using vibration to determine wear failure in high-speed tapping. In [31], a dynamic (mathematical) model for lateral and torsional/axial vibrations in tapping operations was developed. The model was tested with an AIIS1045 steel workpiece and a high-speed steel tap. It was found that vibrations affected the errors, due to feed rate and spindle speed and that torsional/axial vibrations were the main source of instability in the tapping process. Cutting tool coatings are a common method of increasing tool life. For example, the effects of different coatings on tool performance, tool wear mechanisms, and tool life were practically tested in a tapping operation on a nodular iron workpiece in [28]. Three different tools were tested: uncoated HSS-PM (High Speed Steel-Powder Metallurgy), coated HSS-PM TiN (titanium-nitride), and coated HSS-PM TiCN (titanium-carbon-nitride). The wear mechanisms of the HSS-PM tool and of the two coated tools were plastic deformation in the first case, and abrasive wear of the coating and localized peeling in the second. The HSS-PM TiCN coating outperformed the other two tools. [33] used an experimental approach to take direct measurements of cutting section forces of parts of the tapping tool. In [34], a similar study was conducted to measure tapping tool temperature variations. Finally, [29] studied cutting speed and helix angle as determining factors of tool wear and failure during internal thread machining. Three types of taps and different helix angles were used to tap C45 steel,[1] which is a standard material for long-term durability tests.

Besides the research focused on understanding the wear process in tapping, industry is pursuing a different goal: the development of reliable solutions for online monitoring of thread quality. In [15], three possible model-based strategies were identified for tapping quality prediction: mechanically

[1] DIN 17200 grade C45 steel is a non-alloy medium carbon steel used as a reference material for live testing according to relevant standards [29].

based models, statistical models extracted from experimental data, and machine learning-based models.

Mechanically based models mainly apply finite element method (FEM) to the tapping tool. For instance, a model was presented in [32] to calculate torque values for tapping tools of different diameters. In [17], an FEM model was proposed for the design of a tool holder, to reduce the risk of tap breakage during the return instant, and to increase the life-span of the tool.

FEM modeling is nevertheless a challenging task and simulations are time-consuming, due to the different geometries of each tooth, and the non-uniform load distribution on the tap, the radial forces of which can cause lateral deflection of the tool. Some experimental-based models have been combined with statistical approaches to circumvent that limitation and to improve model reliability. For instance, in [21], a model of forces on a single tooth was developed and empirically evaluated. Once the actual forces of the process on each tooth are known, the torque and radial forces can be predicted by multivariate regression from the empirical data. Other examples of this mixed solution can be found in [18, 19]. In those cases, the threading performance was studied in cast iron GG25 without coating and in HSS tapping with TiN, TiCN, TiAlN (titanium aluminum nitride), and TiAlN + WC/C (tungsten carbide/carbon) coatings. In both works, an online monitoring system for thread quality based on torque signal was developed, and principal components analysis (PCA) was used for information extraction and as a statistical technique for thread quality classification.

In line with the data-driven statistical models, ML-based models have also been tested in tapping diagnosis, especially supervised algorithms. Typical machine learning (ML) approaches are the decision tree (DT) and the regression tree (RT), as well as combinations of both models, called ensembles. Following this strategy, two objectives were addressed in [5] for a forming tap operation. First, an experimental measurement of tap wear during threading of cold-forged micro-alloyed HR45 steel was proposed, as abrasive and adhesive wear mechanisms on tap lobes were identified. HSS taps with TiN coating under dry and lubricated conditions were used to perform the experiments. Then, several ML methods for predicting form-tap wear were tested, and a random forest (RF) ensemble without pruned regression trees was selected on the basis of its accuracy as the most appropriate model. [2] also suggested using an ML approach to process quality monitoring, treating the process as a binary classification task [30]. An RF ensemble was also proposed for classification, based on the current profile, to detect out-of-control behavior in a ductile cast iron (GGG50) workpiece tapping process using TiN-coated HSS taps. Finally, [15] pointed out that, in comparison with other machining processes such as milling and turning, the tapping process data showed significant variance, due perhaps to non-deterministic tool wear patterns. It was concluded that this over dispersion in the observed behavior of the tool meant that the automation of quality prediction would be of great value. The same authors also demonstrated the capabilities of ensemble classifiers especially designed for imbalanced datasets, such as boosting and bagging DT ensembles combined with the Synthetic Minority Over-Sampling Technique (SMOTE), and undersampling, among other ML algorithms, to predict thread quality, using torque and force signals as inputs.

All these solutions demonstrate the capability of ML techniques to generate reliable models for the prediction of thread quality under laboratory conditions. Models can also be used to predict the high variability of thread quality due to tapping tool wear. However, a new limitation under industrial conditions also has to be overcome: both human and time costs derived from manual classification of threads are too high and only a low rate of threads are labeled, mainly in binary terms: either failure or no-failure. Conventional ML techniques such as supervised learning (SL) algorithms are not able to deal with unlabeled instances in the datasets, and any such instances should be deleted from the dataset. But a novel family of ML techniques, called semi-supervised learning (SSL) algorithms, includes strategies to extract information from unlabeled instances, improving the accuracy of final prediction models.

In this research, an initial application of SSL to thread quality prediction is presented, considering tapping tool wear, in which the capability of the algorithms to improve upon traditional SL solutions is evaluated. A road map is also shown for the implementation of highly accurate and reliable ML solutions in industrial tapping processes. Figure 2 shows a graphic illustration of the steps proposed to obtain an SSL quality diagnostic solution for a tapping process. To the best of our knowledge, this is the first time a semi-supervised ML approach designed for fault detection and diagnosis during threading operations has been evaluated.

The rest of this article is organized as follows. In Section 2, the semi-supervised approach to machine learning is introduced and the algorithms with the best results are briefly described. In Section 3, the experimental tapping setup, the data collection and treatment procedures, and the datasets that were generated are all described. The results obtained using SSL approaches are then presented in Section 4 and compared with those obtained using SL approaches. Finally, the conclusions and future research directions are presented in Section 5.

## 2 Semi-supervised learning (SSL)

SSL is described in this section. First, a brief overview of this learning approach and a description of its main features

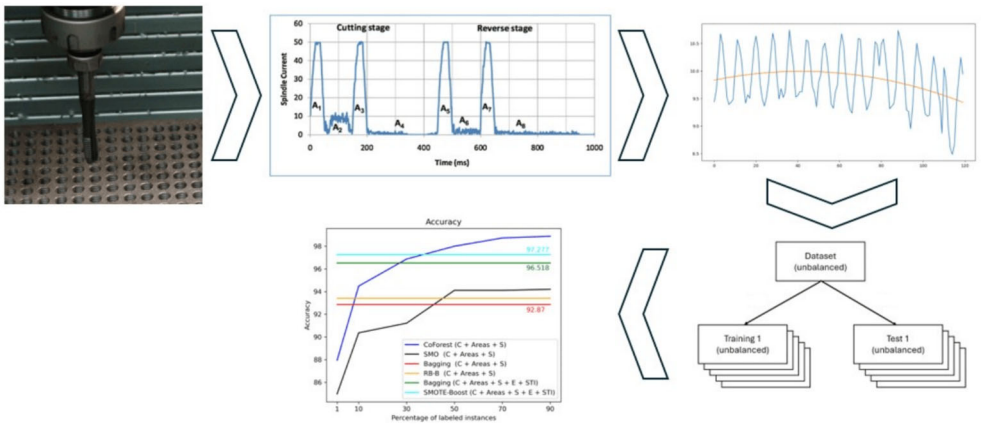


**Fig. 2** Processes and steps for performing a semi-supervised learning (SSL) diagnosis of threading quality

are presented. Knowledge Extraction based on Evolutionary Learning (KEEL) [38], an open source ML software package, is then explained. A more detailed overview of SSL, its main methods, and features can be found in a review [40]. The authors categorized the different SSL methods and proposed a taxonomy. A recent review of the use of SSL methods and techniques for industrial applications can also be found in [35].

## 2.1 Semi-supervised learning (SSL)

The primary objective of ML is to identify patterns and relationships within data so that automated execution of key tasks can be performed, such as detecting abnormal rather than normal behavior, classifying data observations, and optimizing processes based on current behavior. Certain data have to be collected before a process may be automated. The acquisition of machining data is common in many industrial processes, that already have monitoring processes, automated control systems, and other data collection systems. The data, also known as instances or observations, are typically organized into registers, which contain values associated with various measured or calculated features and are collected in a timely manner. For machine learning purposes, the available data are organized into datasets.

ML approaches are classified into four main types [1, 40]: SL, unsupervised learning (UL), reinforcement learning (RL), and SSL. The classification is based on the requirements that the available data must meet and their usage. With SL, the available data must contain the desired ground truth feature(s), such as a measure of tool wear or a classification for a quality measure. Providing these ground truth values is a process commonly referred to as "labeling." When the available data does not contain a ground truth, the term used is "unlabeled" data. UL attempts to discover hidden relationships in the collected data without adding any other information. RL implies that once a model has been learned it can be refined, based on its performance as it is used. Nevertheless, improvements can only be accomplished if reliable data on inaccuracies in the model forecasts can be automatically acquired. Figure 3 shows a perhaps oversimplified graphical representation of the classification of major ML approaches. The figure includes an SSL co-training algorithm as an inductive ML method. For a more complete classification, see [40].

The main drawback commonly cited in SL is the mandatory need for labels for each data instance. Labels must usually be provided by a human, in some cases, an expert. The process is often time-consuming and more expensive than data collection and model learning. Using UL is not a solution because the less information that is available, the less precise the learned model will be, and consequently, fewer models, methods, and approaches may be used to exploit the available data. Alternative methods have been developed to address these issues by integrating SSL and UL techniques, such as active learning (AL) and SSL. On the one hand, AL attempts to create a model with minimal ground truth information, thereby reducing the time and cost of its acquisition. Some of the available unlabeled observations are sent to an oracle, usually an expert, to be labeled. Then, a model is built and tested. If the desired level of accuracy is not achieved, more unlabeled observations are selected and sent to the oracle. The process is repeated until the model is trained to the desired level of accuracy. Different AL approaches primarily stem from the criterion used to select unlabeled instances for labeling. However, this approach still requires the availability of an oracle.

In contrast, SSL uses a dataset containing a small number of labeled instances and a large number of unlabeled instances. The objective of SSL is to leverage unlabeled instances to improve model accuracy, which could only be achieved by using the available labeled instances. SSL methods are based on two different approaches [40]: transduc-

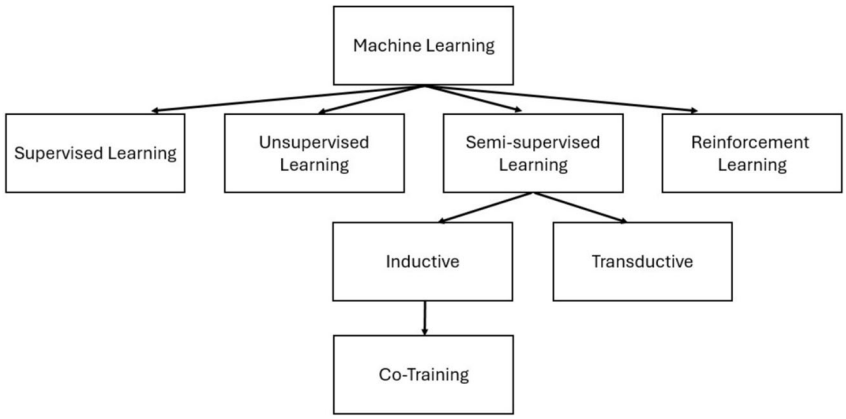


**Fig. 3** Simple classification of machine learning (ML) approaches and methods. Co-training is classified as a semi-supervised learning (SSL) inductive method

tive and inductive learning. Both approaches use unlabeled instances to improve the learning model, but in different ways. Transductive learning focuses on obtaining the best possible labels for the unlabeled instances in the dataset, and it often relies on graph-based methods. It will not therefore always yield an accurate model. Inductive learning, on the other hand, usually yields a model that can be used to label new, unseen instances. Different inductive learning methods rely on different assumptions, leading to a classification of methods based on these assumptions. For more information on the ideas behind SSL and the differences between the methods, see [35, 40]: transductive and inductive learning.

Although the main goal of this research was to study the capabilities of SSL algorithms for the forming problem, some learning approaches had to be discarded at the beginning of the research. The limited number of observations available for training meant that certain learning approaches and models, such as neural networks, which typically require large datasets to generate accurate models, could not be tested. Our approach is somewhat the opposite, as it involves gathering a large labeled dataset versus having a minimum set of labeled observations and many unlabeled observations. Also, other typical approaches to SSL had to be discarded, such as transductive approaches, because the inherent goal of a classification problem is to classify new, unseen observations, and transductive approaches require training with the whole dataset (including validation instances). Finally, this study was focused on the algorithms available in KEEL, so every algorithm or learning approach was obviously not included. For example, the well-known SSL algorithm S3VM [4, 12] is not implemented in KEEL nor are any of the different SSL versions for Fisher Linear Discriminant Analysis (LDA) included [37, 39].

Finally, it is necessary to outline that although using SSL has very interesting benefits, it also has some drawbacks. One benefit is the reduced number of labeled observations. Another benefit is the capability to extract information from unlabeled observations, which can improve the model (compared to the model obtained using only the labeled observations in the dataset). However, the main drawback of using a reduced set of labeled observations with some unlabeled observations is that the result is highly dependent on the number and perhaps the specific labeled observations. Depending on the SSL approach, mistakes in pseudo-labeling unlabeled observations can lead to poor learner model performance that can be worse than a model trained using only labeled observations.

## 2.2 Knowledge Extraction based on Evolutionary Learning (KEEL)

KEEL is a tool that includes several implementations of different ML algorithms [38] and different algorithm types. Some examples are evolutionary and soft computing techniques, for approaching typical ML problems such as classification and regression. Mainly, three different types of ML algorithms are included, grouped into modules such as SL, SSL, methods for learning from unbalanced datasets, and algorithms for data pre-processing.

The set of SSL algorithms includes the following: Self-training [42], co-training [6], tri-training [45], Democratic Co-learning [44], Co-training by Committee (CoBC) [23], Co-Forest [26], ADE-CoForest [14], RASCO [41], Rel-RASCO [43], CLCC [24], APSSC [22], SETRED [25], DE-Tri-Training [13], and various regression types such as LDA and logistic. Versions of the basic supervised methods—the C45 implementation of a DT, Nearest Neighbor (NN), Naive Bayes (NB), and Support Vector Machine (SVM)—can also be used with semi-supervised datasets.

Several of the ML algorithms, such as co-training, tri-training, RASCO, or REL-RASCO, use one or more basic learning algorithms (C45, NN, NB, and SVM). As the basic algorithms are selectable, there are a number of possible combinations of basic algorithms that can be tested for the same SSL method. In addition, the CoBC algorithm can be trained

using either the Adaboost or Bagging learning algorithms, each of which also requires a training base algorithm.

Two different experiments can be defined with KEEL using cross-validation: a ten-fold cross-validation and a 5 × 2-fold cross-validation. In the first, the dataset is randomly divided into ten equal parts, and ten experiments are run. In each experiment, the model is trained using 9 of the 10 parts, and the remaining part is used for testing. A different part is used for testing each time, and the results are averaged. In the 5 × 2-fold cross-validation, five different datasets are randomly created, each containing half of the instances. The model is then trained on one half and tested on the other. The results are averaged over the five datasets.

### 2.3 Brief description of SSL algorithms

Descriptions of the various semi-supervised learning algorithms can be found in the cited references for each algorithm. Several SSL algorithms are also described in recent review articles by [35, 40]. The following only describes the algorithms that achieved the best results for the different datasets, mostly versions of the co-training approach for SSL.

As noted in [6], co-training relies on a dataset containing enough information to define two independent subsets of features ($X_1$ and $X_2$) by splitting the dataset. Two learning algorithms ($A_1$ and $A_2$) are then trained, with each algorithm using the labeled instances of a view. The unlabeled instances are predicted using both models. A reduced number of unlabeled instances predicted with sufficient confidence by one algorithm are "pseudo-labeled" and added to the other algorithm's view. This process is usually repeated until either a loop limit set by the parameter $k$ is reached or there are no more unlabeled instances. An underlying assumption of co-training is that each view can be used to train a good enough model. The co-training algorithm implemented in KEEL is a variation of the above-described algorithm. Three classifiers can be selected. The use of two is as described above, and the third is for the computation of an accuracy metric at each iteration.

The co-training method relies on two key points. First, the confidence of the obtained label for an unlabeled instance has to be computed or estimated. Second, multiple independent views of the same dataset have to be generated while ensuring that sufficient information is contained to construct multiple accurate classifiers.

A similar practice is followed in the tri-training algorithm [45]. Three classifiers are trained. At each iteration, if two of the three unlabeled instances agree upon their prediction, the other unlabeled instance is pseudo-labeled for a particular algorithm. However, instead of assuming that there are enough redundant views of the features or that there are different learning algorithms, tri-training achieves diversity in a different way by using a bootstrapping process to train the initial models, and it then iteratively adds more pseudo-labels to each specific labeled dataset when the label predictions of the other two models agree. In doing so, there is no need for confidence evaluation with tri-training.

CoBC [23] is another algorithm that uses a co-training approach. It uses a two-view approach to improve results, but it combines the co-training approach and tree-based ensembles. CoBC is best suited for datasets with four specific characteristics: (1) sufficient redundant views can be defined, (2) there is a large number of classes, (3) there are relatively few labeled instances, and (4) there is a relatively large number of unlabeled instances. CoBC uses a statistical approach to consider the probabilities of the intermediate outcomes of the internal nodes of the trees, rather than simply traversing the ensemble trees to obtain a prediction. In that way, both the path to the selected leaf node and the other paths can contribute to the predicted class. The KEEL CoBC implementation can use an Adaboost boosting method and Bagging, a bootstrapping method, both of which are ensembles that can be implemented in KEEL. Both require a base classifier, so C45, NN, and SMO can be chosen as base classifiers for the ensembles in CoBC.

Democratic co-learning [44] avoids the need for redundant and independent feature sets by training different algorithms instead of having multiple views. Different algorithms are expected to have different inductive biases even when trained on the same data. The predicted label is obtained by means of weighted voting of the classifiers, and the pseudo-labeled instance is added to the training sets of the classifiers whose predictions are unlike the majority predictions.

Co-Forest [27], a variation of the co-training algorithm, employs RF [7] to identify the most confident instances for pseudo-labeling. RF is an ensemble, $H^*$, of $n$ classifiers, rather than two or three classifiers, with other co-training-based approaches. In the ensemble $H^*$, the confidence of an unlabeled instance for a classifier, $h_i$, is determined by computing the degree of agreement on labeling with the other classifiers. Therefore, the set of classifiers, $H^*$, are firstly trained in Co-Forest using the labeled instances, and then the unlabeled instances that the other classifiers have previously selected are labeled to refine each classifier, $h_i$, in $H^*$.

DE-Tri-Training [13] (tri-training with data editing) is an SSL clustering algorithm that makes use of the tri-training SSL algorithm. Firstly, it uses the labeled instances to increase the labeled set. The Nearest Neighbor Rule [11] is used to improve the quality of the increased labeled set, to reduce the influence of noise and misclassified instances. The labeled instances are then used for a Seeded-K-means and Constrained-K-means algorithm [3].

### 2.4 Additional basic algorithms with the KEEL co-training method

The KEEL co-training implementation offers a choice between four basic learning algorithms that can be combined using the co-training method, namely C45, SMO, NB, and NN. These well-known basic algorithms use different approaches to machine learning, and many of them, although relatively simple compared to other learning algorithms, achieve reasonably good results on many datasets. However, the idea of applying the co-training approach to other generally more complex learning techniques and approaches has led to proposals such as tri-training, Co-Forest, CoBC, and democratic co-learning. Based on the above approach, the co-training implemented in KEEL was therefore extended, as the use of more complex basic algorithms might have improved the results when using the original co-training method.

Three more complex learning algorithms were added as basic algorithms for use in the KEEL SSL co-training method, to test whether an improvement can be achieved by using more complex basic learners in co-training. These are a logistic regression classifier (LOG in the results tables) and two ensembles that use the C45 algorithm as the base classifier. Although they do not appear in the results tables, AdaBoost, a boosting method, and Bagging were added to the set of learning approaches.

Implementations of those new basic learning algorithms for co-training were already included in KEEL and were adapted, so that they could be used with the co-training method.

### 2.5 SVM: a reference SL algorithm for comparison

The capabilities of SSL techniques must always be compared with the performance of an SL algorithm to identify any improvements over conventional and simpler ML solutions. For instance, [15] proposed the following SL algorithms as base algorithms for comparative purposes: $k$-NN (with $k = 3$), multi-layer perceptron (MLP), SMO, Tree U (a J48 implementation of a non-pruned DT), Bagging, and RF.

Therefore, the SSL datasets were tested in this research with the SSL version of the basic SL algorithms implemented in KEEL for comparative purposes: C45SSL, NBSSL, NNSSL, and SMOSSL. All four algorithms are simple adaptations of the supervised C45, NB, NN, and SMO algorithms for use with semi-supervised datasets where unlabeled instances are first removed, and then the algorithms are trained using only the labeled instances, so that the same semi-supervised dataset may be used as input for training the supervised algorithms. The C45 instead of the C45SSL algorithm was selected for the following task, and the same applies to the other basic algorithms. Finally, the algorithm selected in this research for comparative purposes was the KEEL SMO algorithm for SSL datasets (SMOSSL), as discussed in Section 4.1.

### 2.6 Experimental design

The experimental design was focused on varying the available algorithms more than on finding the optimal parameters for the algorithms that were tested. Variations were incorporated, based on the nature of the SSL algorithms and the complexity of their configurations. In KEEL, one or more base learning algorithms may be selected for use with several SSL algorithms. For example, three base learning algorithms may be selected for KEEL's co-training algorithm: one for each view and one for evaluating the results. tri-training, CoBC, and others also require one or more base learning algorithms. Thus, changes to the base classifiers used in the configuration of the SSL algorithm meant that approximately 400 algorithms and combinations of algorithms could be evaluated in this research. Additionally, some base learning algorithms, such as the SMO algorithm, require considerable training time. So, extensive tuning of algorithms parameters was not considered a priority in this research, and the KEEL algorithms were run using their default parameter values that were expected to produce good results.

## 3 Experimental setup and dataset description

In this section, the experimental setup and the data collection procedure are introduced, outlining the similarities between this procedure and real industrial conditions for tapping. The original experimental dataset is also presented. Secondly, the plain semi-supervised datasets, generated from the original experimental dataset to simulate an extensive range of real industrial conditions, are presented. Thirdly, the extended semi-supervised datasets, based on the application of the SMOTE technique to the plain semi-supervised datasets, are presented to simulate balanced conditions between failure and non-failure conditions, a key requirement for a proper evaluation of SSL solutions with different levels of unlabeled instances. Then, the datasets were improved considering a sliding window, to take into account that under industrial conditions each tap is correlated with the measurements of some previous tapping operations (historical data on the same tapping tool). Minor improvements to the datasets, due to a measurement drift in the experimental signals, are proposed in the last two subsections.

### 3.1 Experimental setup and data collection

The experimental data sheet was collected using a computer numerical control (CNC) under high-speed cutting

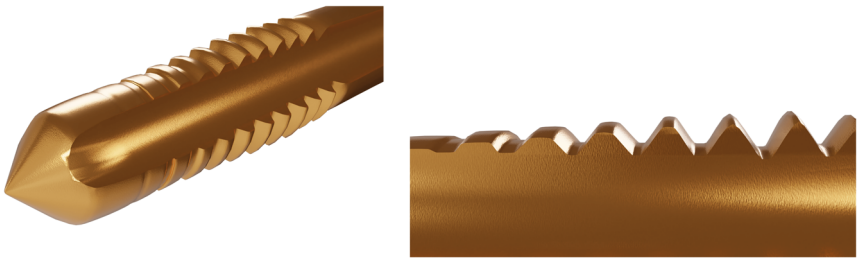

**Fig. 4** Geometry of the tap tool

(65 m/min) and dry conditions while executing a tapping cycle. A nodular cast iron (GGG50) workpiece was used. The tapping process required previous drilling operations. All holes were inspected according to the ISO 2 requirement (6 H). The tapping tool has three flutes to sculpt M10 x 1.5 mm metric threads, as can be seen in Fig. 4.

Five different coatings were selected as can be seen in Table 1, because each coating has different mechanical properties to improve the cutting process. Moreover, a go-no-go gauge (6 H) was selected to inspect the quality of all the threads. If the go side of the gauge could not pass through the thread, then the thread was considered faulty. The reason for that evaluation strategy was directly related to one of the advantages of go-no-go gauges: quick and efficient verification of thread specifications. The gauge tolerance was lower than 10% of the product tolerance, fixed by the ISO 1502 standard for screw threads and test gauge instruments.

The results of the experimental tests are summarized in Table 1. The first column shows the five tap-tool coatings. The second column shows the number of tap tools with each kind of coating. The third column shows the total number of sculpted threads *per* type of coating. The fourth one shows the percentage of acceptable threads that passed the inspection, and the last column shows the percentage of unacceptable threads that were not within the set tolerance margins.

For each thread, torque signals were recorded. Figure 5 illustrates the torque signals extracted from the internal drive device of the spindle motor. The blue and red profiles are the first acceptable and unacceptable threads, respectively. Both signals have the same stages, the first one is the cutting zone when the thread is sculpted, and the second stage is the reverse zone when the tap returns to the initial point, as can be seen in the movements of the tap in Fig. 5.

The thread profile was divided into seven areas (A1–A7) to characterize the tapping operation. The torque signal throughout a tapping cycle was divided into two stages to understand the physics of each area. The first is the cutting stage when the tap tool sculpts the thread profile from the initial to the final positions, respectively. This zone is divided into four torque areas. Area A1 represents the torque evolution area of the main spindle motor during the acceleration period. Area A2 is the torque evolution of the main spindle motor while the threads are cut with the tap. Area A3 is the torque evolution in the deceleration stage. Once the tap base is at the final point, there are always feedback signals to both motor regulators. As the CNC is programmed to maintain synchronism between both motors, minute shifts will occur, leading to friction torque, recorded in Area A4.

In the reverse stage, Area A5 represents the torque evolution during the tap acceleration phase to invert the spindle rotation when beginning to exit the hole. A6 is the tap torque evolution area during the tap reversal phase. Finally, the deceleration torque is represented in Area A7. Graphs of the evolution of A1–A7 for each tap can be found in the Supplementary Material, along with the dataset.

The features and their variation range in the dataset are summarized in Table 2. The coating and tool ID features were coded as numbers. The thread number was the sequence of threads that each tap sculpted, before it was rejected, due to significant wear. Tool ID is the number assigned to each tap tool. The last column shows the range of each experimental feature in the datasheet.

A more detailed description of the tapping process, the physical machines and the electronic devices, a description of the signals, the method of obtaining the measurements, data pre-processing, and data curation can be found in [15].

**Table 1** Contents of the experimental results

| Coating | Tools | Number of threads | Go threads | No-go threads |
|---|---|---|---|---|
| TiN | 15 | 2290 | 77% | 23% |
| Steam | 7 | 1751 | 86% | 14% |
| TiNC | 5 | 1148 | 91% | 9% |
| AlCrN | 5 | 1519 | 89% | 11% |
| TiN + Steam | 4 | 407 | 92% | 8% |
| Total | 36 | 7115 | 85% | 15% |

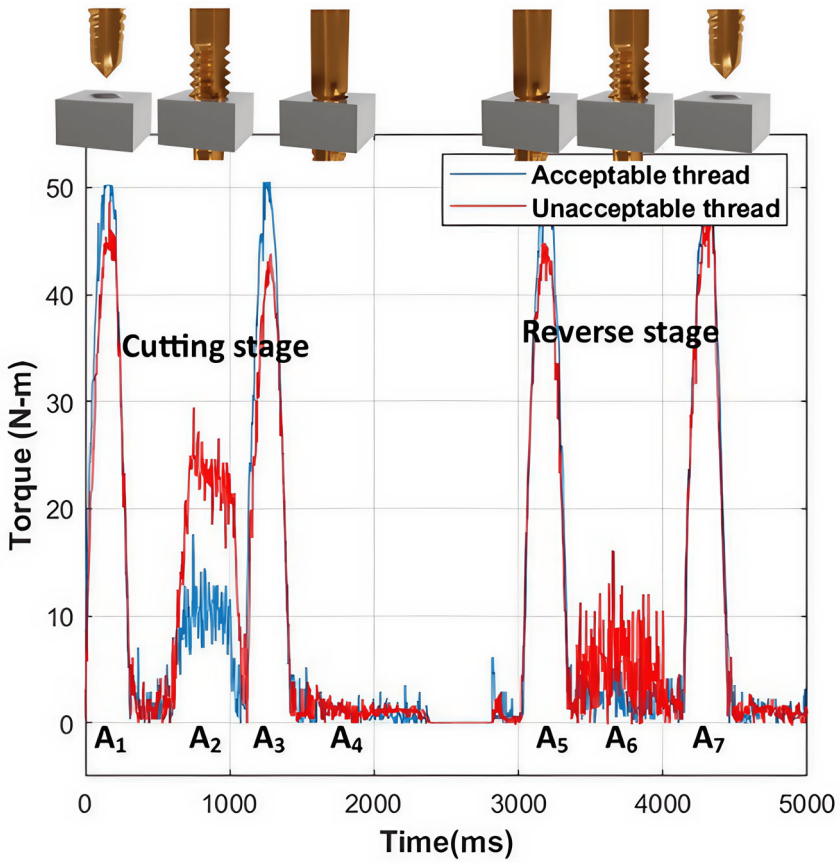


**Fig. 5** Torque evolution during acceptable and unacceptable thread cycles, respectively

This procedure is well-established in the state of the art; for instance, the same data acquisition and signal processing procedures have also been followed by other authors [2].

Feature importance was studied using the permutation importance tools available in the Python package scikit-learn, with a random forest base classifier. The results are shown in Fig. 6. Features with lower importance are closer to 0. According to the figure, features corresponding to Areas 2, 4, and 6 were the most important for classification. This result was predictable, because there is lengthy contact between the tap tool and the material (direct thread, reverse thread, and friction torque stage) in all 3 areas during which time most information on the tapping process may be collected.

**Table 2** Features and ranges of experimental data

| Feature | Abbreviation | Range |
|---|---|---|
| Coating ID | Coating | 1–5 |
| Tool ID | ToolID | 1–36 |
| # Thread | number | 1–519 |
| Torque signal area A1 | A1 | 0.0–11.31 N*m*s |
| Torque signal area A2 | A2 | 0.0–9.87 N*m*s |
| Torque signal area A3 | A3 | 0.0–11.10 N*m*s |
| Torque signal area A4 | A4 | 0.0–17.34 N*m*s |
| Torque signal area A5 | A5 | 0.0–11.73 N*m*s |
| Torque signal area A6 | A6 | 0.0–6.41 N*m*s |
| Torque signal area A7 | A7 | 0.0–10.92 N*m*s |
| Thread quality inspection | go, no-go | 1–2 |

### 3.2 Plain semi-supervised datasets

In an SSL setting, some of the available instances, usually only a few, will be labeled, while most of the instances will be unlabeled. To approach this situation, a training and testing process was performed using files in which some of the labels of the instances in the training files had been removed. Different percentages of labeled and unlabeled instances were used to train the models. The test files were fully labeled.

Using a 5 × 2-fold cross-validation approach, the initial dataset should be randomly divided more or less equally into a pair of training and test files (Fig. 7). Bearing in mind that the number of tapping tools, and hence the number of instances, is not the same for each coating, some care should be taken to maintain the proportion of instances for each coating in both training and test files. Otherwise, the initial problem of imbalance may be magnified, accentuating or ren-

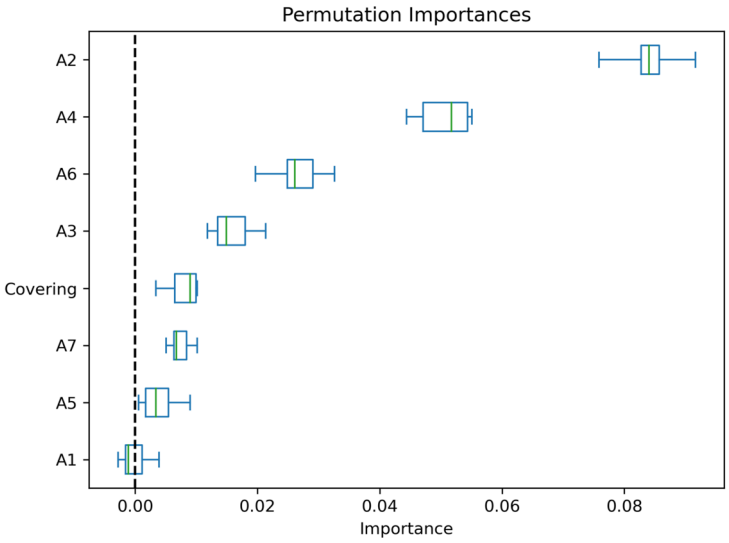


**Fig. 6** Estimated importance of the features for classification

dering nearly impossible the problem of obtaining accurate models, especially if the most unbalanced files were used for training. Five different pairs of training and test files should be generated, and the final result is the averaged results of the five test files.

Several datasets were generated, in order to assess the importance of the number of labeled instances in the training files and the improvement that the unlabeled instances can provide. Each one maintained a certain percentage of labeled instances in the training files (the same percentage in the five training files). Again, due to the unbalanced nature of the original dataset, special care should be taken when selecting the instances to be unlabeled. As the less represented class is the no-go (fail) one, in general, at least one no-go (fail) instance was kept for each tapping tool in the training files. It should be said that this constraint makes the training files with fewer than 30% of labeled instances somewhat unrealistic, as the effect of the constraint can actually be seen as a process of "balancing" the datasets. When 30% or more of the labeled instances were kept in the training files, the "balancing" effect almost disappeared. In addition, the same seed was used to generate the random number sequences, to assess whether increasing information improved the model or not, so that a file with a higher percentage of labeled instances contained the labeled instances in a file with a lower percentage of labeled instances, plus some new ones.

The different percentages used for learning and evaluating the models are shown in Table 3. The first column shows the percentage of labeled instances in the training files, ranging from 1 to 90%. The second and third columns contain the number and percentage of Go (ok) and No-go (fail) labeled instances in the training files. The fourth and fifth columns contain the number of Go (ok) and No-go (fail) labeled instances in the test files. As can be seen from the percentages in the third column of the table, the number of no-go instances labeled for the 1% and perhaps 10% training files was higher than in the other files, due to the imposed constraint of having at least one no-go labeled for each tapping tool.

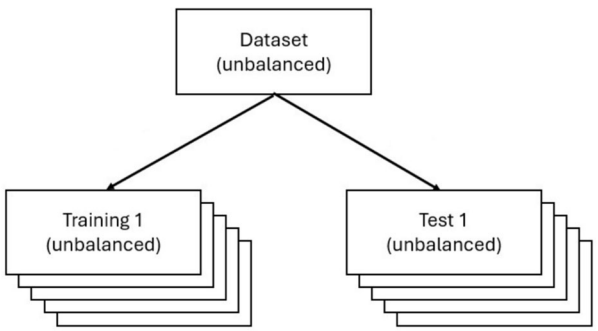


**Fig. 7** In a 5 × 2 cross-validation, the original data file is randomly split into two files, one for training and one for testing. Five different splits are performed

**Table 3** Percentage of labeled instances and number of labeled and unlabeled instances in training and test files

| Labeled Instances | Semi-supervised Training | | Test | |
|---|---|---|---|---|
| | Go (ok) | No-go (fail) | Go (ok) | No-go (fail) |
| 1% | 41 | 35 (85.36%) | 3026 | 520 |
| 10% | 306 | 61 (19.93%) | 3026 | 520 |
| 30% | 913 | 163 (17.85%) | 3026 | 520 |
| 50% | 1528 | 274 (17.93%) | 3026 | 520 |
| 70% | 2131 | 376 (17.64%) | 3026 | 520 |
| 90% | 2738 | 481 (17.56%) | 3026 | 520 |

Once the unlabeled instances were selected, the features ToolID and Thread Number were removed and the actual files for training and testing contained only 9 features: CoatingID, A1–A7 torque signal areas, and thread quality. The quality value of the unlabeled instances was replaced with the KEEL key value "unlabeled." All the features were scaled down (or up) to values between 0 and 1, as there can be significant differences within the ranges of the different features, and some learning algorithms are very sensitive to the magnitude of the different features.

However, as observed in [15], more than one dataset was generated, with different features. The different datasets are shown in Fig. 8 and described in the following subsections.

### 3.3 Extended semi-supervised datasets: SMOTE

In [15], it was noted that the imbalance in the number of instances *per* class might have a detrimental effect on the learned classifiers, and two additional approaches were used to mitigate that effect. The first approach was to use classical rebalancing strategies: oversampling, or increasing the instances of underrepresented classes, and undersampling, or reducing the instances of overrepresented classes.

Basically, the oversampling strategy consisted in the use of SMOTE-based techniques [8]. However, oversampling can be applied in several ways. SMOTE was applied to the plain dataset by generating several different numbers of new instances for the less represented class (increasing them by 100%, 300%, 500%, or as many new instances as needed, so that the classes were equally represented). But some techniques, such as boosting, have their own approach. In summary, in [15], Bagging + SMOTE (with different number of instances generated), Bagging + Random Undersampling, Bagging + Random Balance [10], RAMOBoost [9], RUSBoost [36], and RB-Boost [16] rebalancing techniques were applied, depending on the learning algorithm.

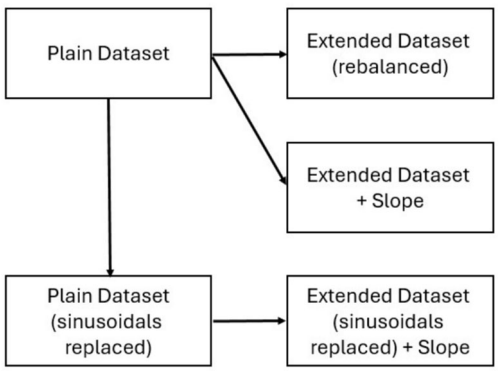


**Fig. 8** The original data file generates five different datasets with slightly different information

Furthermore, in [15], rebalancing techniques were used in conjunction with the different datasets that were generated: both the simple dataset and the extended semi-supervised dataset with the slope of the torque signal areas (A1–A7) on a sliding window, as described below.

The same rebalancing approach was modified when used in conjunction with the SSL approach, due to several problems. First, although SMOTE is implemented in KEEL, the algorithms in the KEEL SSL module take no account of unbalanced datasets. Furthermore, in SSL datasets with fewer labeled instances, it can be difficult to generate new instances with sufficient variance to be useful for learning purposes. The same can be said of undersampling strategies, as removing instances when there are too few labeled instances hardly appears to be a very good idea, so the undersampling techniques were discarded. The cross-problem of SSL with unbalanced datasets is not addressed in KEEL.

New datasets were therefore generated for testing purposes, using the SSL plain datasets and the SMOTE algorithm implemented in the Python programming language, without changing the default values for the parameters of Python's SMOTE algorithm.

### 3.4 Extended semi-supervised datasets: sliding window

In [15], in addition to using just the plain dataset in its SL setting, data enhancement was also suggested through the use of a sliding window and including, as new columns, the slope of a linear regression computed using the data in the sliding window for each of the torque signal area features (A1–A7) in the dataset. The best results were obtained using a sliding window of length 11. Indeed, in [15], four datasets were used: the plain dataset, and using a sliding window, a dataset extended with the slope of the linear regression, a dataset extended with the slope and error of the linear regression, and finally a dataset extended with the slope and standard error of the linear regression and the True Strength Index (TSI).

Following the same path as in [15], a second dataset was therefore generated using a sliding window of length 11, computing the slope of the linear regression in the sliding windows, and adding the seven slopes to the available instances. The slopes were separately computed for each tool. The first 10 values of each tool sequence of threads meant that the slope could not be computed and the data were therefore removed from the datasets. Obviously, the number of dataset instances was reduced accordingly. Again, after the sliding windows were generated, the linear regression was computed, the unlabeled instances were selected, and the ToolID and Thread Number features were removed, until the actual training and test files contained 16 features: Coat-

ingID, A1–A7 torque signal areas, S1–S7 slope of the linear regressions for the torque signal areas, and thread quality. Neither the linear regression error nor the TSI values were computed. For the unlabeled instances, the quality value was replaced by the KEEL key value "unlabeled." Since there can be a noticeable difference in the ranges of the different features, and some learning algorithms are very sensitive to the magnitudes of the different features, they were all scaled to values between 0 and 1.

### 3.5 Extended semi-supervised datasets: A1, A3, A5, and A7 sinusoidal signals replaced by linear regressions

Two other datasets were also generated for the SSL approach. The torque signal areas A1, A3, A5, and A7 in the dataset were sinusoidal signals. However, when using an SSL approach, labeled instances provided only a few reference points in the sinusoid. Therefore, these few available points in the sinusoidal signals could have hidden the trend and could have been interpreted by the learning algorithm as noise in the signal.

Usually, the sinusoidal signal initially varied in a fairly flat range, and at some point, a downward trend appeared (Fig. 9). A linear regression was calculated using a 2nd degree polynom as shown in Eq. 1. The regression output of a 2nd degree polynom followed the trend observed in the signal better than a 1st degree polynom.

$$A_n = b_0 + b_1 \times x + b_2 \times x^2 \tag{1}$$

Therefore, the A1, A3, A5, and A7 signal values were first replaced by their calculated linear regressions. Then, the process of generating the SSL datasets was repeated. The unlabeled instances were selected, and the features ToolID and Thread Number were removed, so the training and test files contained only nine features: CoatingID, A1–A7 torque signal areas, and thread quality. For these unlabeled instances, the quality value was replaced with the KEEL key value "unlabeled." Finally, all features were scaled to values between 0 and 1.

### 3.6 Extended semi-supervised datasets: A1, A3, A5, and A7 sinusoidal signals replaced by linear regressions + sliding windows

After replacing the sinusoidal signals A1, A3, A5, and A7 by their linear regressions, a sliding window of length 11 was taken and the slope of the linear regression of the instances within the sliding window was calculated.

The seven slopes were added to the available instances. The slopes were calculated for each tool separately, as the slopes cannot be calculated with the first 10 values of each tool, which were therefore removed from the datasets. After calculating the slopes of the linear regression for each sliding window and selecting the unlabeled instances, the features ToolID and Thread Number were removed, and the actual training and test files contained 16 features: CoatingID, A1–A7 torque signal areas, S1–S7 slope of the linear regressions for the torque signal areas, and thread quality. Neither the linear regression error nor the TSI value was calculated. For the unlabeled instances, the quality value was replaced by the KEEL key value "unlabeled." Finally, all features were scaled to values between 0 and 1.

## 4 Results and discussion

In this section, the results obtained using the SSL methods and algorithms on the different datasets that were generated for different percentages of labeled instances are presented, discussed, and compared with other authors. In [15], the same dataset was used, but with an SL approach. The results obtained with the SSL techniques and methods were compared to those obtained with the SL approach. The goal, of course, was to obtain similar results or, if possible, better results using an SSL approach. However, it should be noted that in [15], several SL algorithms were tested on the simple (class unbalanced) dataset, and two additional approaches

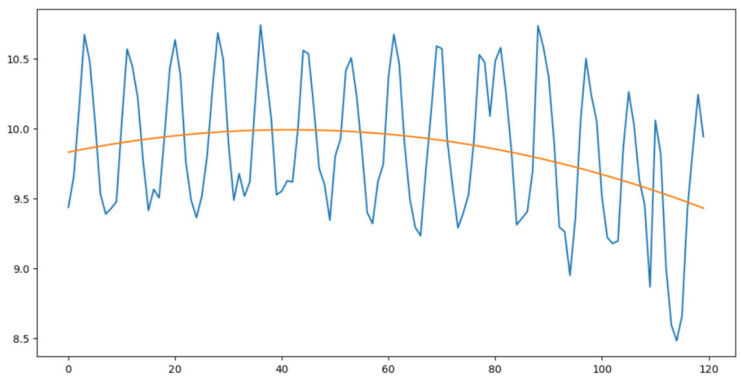


**Fig. 9** Area 3 of ToolID 25 original signal (in blue) and linear regression computed (in orange)

**Table 4** Percentage accuracy levels of the different supervised algorithms

| Labeled | Supervised methods | | | |
|---|---|---|---|---|
| Labeled instances | SMO | C45 | NN | NB |
| 1% | 86.55% | 84.56% | 82.07% | **88.14**% |
| 10% | **89.98**% | 88.72% | 88.17% | 89.26% |
| 30% | **90.31**% | 89.79% | 88.38% | 89.38% |
| 50% | 90.33% | **90.54**% | 88.96% | 89.42% |
| 70% | **90.56**% | 90.53% | 89.17% | 89.42% |
| 90% | 90.68% | **91.13**% | 89.36% | 89.31% |
| 100% | **90.77**% | 90.64% | 89.40% | 89.36% |

Each row represents the results, using a different percentage of labeled instances, ranging from 1 to 100%. The first column shows the dataset percentage of labeled instances. The remaining columns are the accuracy of the different supervised learning (SL) algorithms in KEEL. The best result for each row is shown in bold

were tested in an attempt to improve the results by modifying the initial dataset. Namely, the intention was to balance the number of instances of the different classes using SMOTE and to increase the number of dataset features.

### 4.1 Plain semi-supervised dataset

Firstly, the performance of the supervised learning algorithms available in KEEL on the plain semi-supervised datasets was tested for comparison with previous works. The test results are shown in Table 4. The first column shows the percentage of labeled instances, from 1 to 100%. The other columns show the results obtained for each of the SL algorithms. The best results for each percentage of labeled instances in the dataset are shown in bold.

After observing the results shown in Table 4, it was decided to use the SMO algorithm as a reference. It obtained the best score for accuracy more frequently than the sum of the others, showing a clear progression in its accuracy score as the number of labeled instances in the dataset increased, and it was close to the highest score obtained by the C45 algorithm in two datasets where over 50% of the instances were labeled..

Once the SL algorithm that will serve as the baseline had been chosen and its accuracy evaluated, the SSL algorithms were tested on the plain semi-supervised datasets. The test results are shown in Table 5. The first column shows the percentage of labeled instances, and the second column shows the results obtained using the SL reference algorithm (SMO). The following columns show the SSL algorithms that had the best results for each percentage of labeled instances within the datasets.

As Table 5 shows, the results ranged from 63.02 to 91.53%. The SL reference algorithm achieved an accuracy level of 89.98% with the dataset of 10% labeled instances, and it hardly improved the result, even with the dataset of 90% labeled instances. The SSL algorithms hardly improved the results of the reference SL algorithm. An accuracy level of 91.53% was achieved using the 90% labeled dataset, which was only a slight improvement over the SL algorithm. Only in the case of very few labeled instances (1% dataset) did the accuracy of most models drop significantly. It is interesting to note that different combinations of base algorithms for the co-training and tri-training algorithms yielded the best results.

For comparison, in [15], using a 100% of labeled instances and the plain dataset, the accuracy of the most basic classifier, a DT, was 90.683%, the MLP yielded an accuracy of 91.667%, and a Bagging algorithm, with an accuracy of 92.229%, yielded the best performance. These accuracies are in the same range as those obtained in initial first tests and shown in previous tables.

**Table 5** Accuracy of the different semi-supervised learning (SSL) algorithms versus SMO algorithm in percentage

| | Supervised | Semi-supervised methods | | | | | |
|---|---|---|---|---|---|---|---|
| Lab inst | SMO | TT-NB-SMONB | CT-SMO-C45LOG | TT-C45-SMOSMO | TT-SMO-SMOC45 | TT-SMO-C45SMO | TT-C45-C45SMO |
| 1% | 86.55% | **88.37**% | 63.02% | 86.41% | 86.40% | 86.47% | 85.72% |
| 10% | 89.98% | 89.48% | **90.98**% | 90.30% | 90.23% | 90.29% | 89.79% |
| 30% | 90.31% | 89.80% | 90.78% | **90.97**% | 90.94% | 90.96% | 90.55% |
| 50% | 90.33% | 89.74% | 90.88% | 90.99% | **91.01**% | 91.00% | 90.58% |
| 70% | 90.56% | 89.83% | 91.02% | 91.33% | 91.34% | **91.34**% | 91.03% |
| 90% | 90.68% | 89.92% | 90.98% | 91.52% | **91.53**% | **91.53**% | **91.53**% |

Each row represents the results, using a different percentage of labeled instances, ranging from 1 to 90%. The first column contains the percentage of labeled instances in the dataset. The second column contains the results of the reference SL algorithm (SMO). The remaining columns show the best results of the different SSL algorithms for a given dataset. The best result for each row is shown in bold. In the table, $CT$ stands for co-training, $TT$ for tri-training, $LOG$ for logistic

**Table 6** Accuracy of the different algorithms in percentages

| Lab inst | Supervised SMO | Semi-supervised methods TT-NB-NBSMO | TT-SMO-NNNN | TT-NN-SMONN | COBC-ADABOOST-C45 | TT-C45-C45NN | TT-C45-NBNN |
|---|---|---|---|---|---|---|---|
| 1% | 85.48% | **88.31**% | 83.41% | 82.62% | 84.67% | 83.44% | 87.48% |
| 10% | 87.56% | 88.28% | **89.45**% | 89.30% | 87.88% | 88.36% | 88.35% |
| 30% | 88.00% | 88.87% | 89.57% | **89.60**% | 88.75% | 89.37% | 89.05% |
| 50% | 87.77% | 88.56% | 88.99% | 89.01% | **89.19**% | 89.06% | 88.71% |
| 70% | 87.73% | 88.48% | 88.58% | 88.59% | 89.23% | **89.73**% | 89.10% |
| 90% | 87.98% | 88.65% | 88.41% | 88.42% | 89.35% | 89.04% | **89.89**% |

Each row represents the results, using a different percentage of labeled instances, ranging from 1 to 90%. The first column contains the percentage of labeled instances in the dataset. The second column contains the results of the reference SL algorithm (SMO). The remaining columns are the different SSL algorithms that gave the best result for a given dataset. The best result for each row is shown in bold. In the table, $TT$ stands for tri-training

## 4.2 Extended semi-supervised dataset with SMOTE

Balancing the dataset may be a suitable strategy to increase model accuracy. Table 6 shows the results of using the datasets obtained after SMOTE was applied to the plain semi-supervised datasets. Again, the first column shows the percentage of labeled instances, the second column shows the results obtained using the SL reference algorithm (SMO), and the following columns show the SSL algorithms that achieved the best results for each dataset.

As can be seen, the results ranged between 82.62 and 89.89%. The reference SL algorithm with 10% of labeled instances reached 87.56% with the dataset and there was almost no improvement in the results, even when using the dataset with 90% of labeled instances. It is of interest that some combinations of basic classifiers with the tri-training algorithm almost always performed best of all.

Comparing the results in Table 6 with the results in Table 5, it can be seen that some improvement was obtained when using the least labeled dataset, but the reference SL algorithm obtained worse results and the SSL algorithms basically obtained equally poor or worse results, so using SMOTE and an SSL approach may not be a good solution to the problem. A conclusion that might explain why adding the synthetic observations to the original labeled observations decreased the accuracy results of the SSL algorithms.

Generating new synthetic observations from a limited number of labeled observations may not produce examples that represent the underlying data distribution sufficiently well. So, creating new synthetic observations to match the number of class observations might not compensate for the lack of sufficient labeled data. Missing information cannot be replaced, which could lead to bias in the data and that could in turn explain why adding the synthetic observations to the original labeled observations decreased the accuracy of the reference SMO algorithm. The results demonstrated a decrease when new synthetic observations were added to the original labeled observations for all the tested datasets. Conversely, SSL techniques have the potential to extract sufficient information from unlabeled observations to improve the model obtained from labeled observations. The approach shows promise in enhancing the quality of information avail-

**Table 7** Percentage accuracy of the different algorithms

| Lab inst | Supervised SMO | Semi-supervised methods TT-NB-NNSMO | TT-NN-SMONB | TT-SMO-NNNB | TT-NB SMOSMO | DTT-C45-NBSMO |
|---|---|---|---|---|---|---|
| 1% | 76.93% | **86.63**% | 86.25% | 85.98% | 81.56% | 72.25% |
| 10% | 85.33% | 89.10% | **89.12**% | 89.05% | 88.82% | 88.10% |
| 30% | 88.04% | 89.08% | 89.12% | **89.30**% | 88.79% | 86.11% |
| 50% | 87.89% | 89.07% | 89.11% | 89.11% | **89.83**% | 88.73% |
| 70% | 88.57% | 89.31% | 89.35% | 89.37% | **90.06**% | 89.53% |
| 90% | 89.22% | 89.32% | 89.34% | 89.30% | 90.07% | **90.51**% |

Each row represents the results, using a different percentage of labeled instances, ranging from 1 to 90%. The first column contains the dataset percentage of labeled instances. The second column contains the results of the reference SL algorithm (SMO). The remaining columns show the different SSL algorithms that yielded the best result for a given dataset. The best result for each row is shown in bold. $TT$ stands for tri-training and $DTT$ stands for DE-Tri-Training

able for building the model, potentially surpassing the contributions of synthetic observations generated by SMOTE. The accuracy of SSL algorithms may therefore be affected by the incorporation of labeled observations synthetically generated by SMOTE, compared to the results obtained if SMOTE is not used.

Therefore, SMOTE was discarded, as its use together with the plain dataset provided no significant improvement to the results obtained using the plain dataset alone, and it may introduce bias into the dataset, as seen in the performance of the SL reference algorithm.

### 4.3 Extended semi-supervised dataset that includes the linear regression slopes within a sliding window of length 11

As noted in previous works [15], the strategy of a sliding window in this sort of dataset may increase model accuracy, so it was tested. The results of using the extended semi-supervised dataset including the linear regression slope within a sliding window of length 11 are shown in Table 7. The first column shows the percentage of labeled instances, the second column shows the results obtained using the SL reference algorithm (SMO). The following columns show the SSL algorithm that achieved the best performance for each percentage of labeled instances within the dataset.

As can be seen in Table 7, the results rarely improved on those in Tables 5 and 6. The results of the SSL algorithms in no way improved upon previous results.

After examining the results in Table 7, it can be concluded that the addition of the slope, computed using the sliding windows, to the available datasets rather than improving the learning process, actually misled it. The reason may be because of the way that the SSL algorithms work. co-training-based algorithms, such as tri-training, use the labeled instances to train two or more models that are used to compute a label (pseudo-label) for the unlabeled instances in the semi-supervised dataset, in the hope that increasing the number of labeled instances will improve the learned model and improve the accuracy measure of the classifier. However, if the wrong label is chosen for pseudo-labeling, the desired improvement may not be achieved.

Therefore, using a few isolated values of the sinusoidal signals in A1, A3, A5, and A7 might be interpreted as background noise by the learning algorithm. Depending on the labeled instances, there might be no difference or not enough difference in the values for the sinusoidal signals. Worst of all, adding the slope could make the problem worse. Although this experiment did not achieve better results, it did provide a possible reason for the low performance, prompting the search for more suitable data-treatment techniques.

### 4.4 Plain semi-supervised dataset, replacing the A1, A3, A5, and A7 torque sinusoidal signal areas with a linear regression

A linear regression on the signals A1, A3, A5, and A7 was applied to dampen the noisy sinusoidal behavior of the signals, and the performance of the SSL algorithms was tested on the new datasets. Table 8 shows the results of using the plain semi-supervised dataset, replacing the sinusoidal signals A1, A3, A5, and A7 by a linear regression computed from their values. The first column shows the percentage of labeled instances, the second column shows the results obtained using the SL reference algorithm (SMO), and the following columns show the SSL algorithms with the best performance for each percentage of labeled instances in the dataset.

As can be seen from the results in Table 8, better results were obtained by replacing sinusoidal signals with the cor-

**Table 8** Percentage accuracy of the different algorithms

| Labeled instances | Supervised SMO | Semi-supervised methods | | |
|---|---|---|---|---|
| | | TT-NNNNNN | TT-C45C45NN | TT-C45NNC45 |
| 1% | 88.61% | **90.42%** | 86.38% | 86.33% |
| 10% | 92.85% | 95.48% | **95.49%** | 95.10% |
| 30% | 93.85% | 96.94% | 96.76% | **97.01%** |
| 50% | 94.11% | **97.82%** | 97.79% | 97.70% |
| 70% | 94.11% | **98.10%** | 97.91% | 97.93% |
| 90% | 94.20% | **98.30%** | 98.04% | 98.00% |

Each row represents the results when using a different percentage of labeled instances, ranging from 1 to 90%. The first column contains the percentage of dataset labeled instances. The second column contains the results of the reference SL algorithm (SMO). The remaining columns are the different SSL algorithms that gave the best result for a given dataset. The best result for each row appears below in bold. In the table, *TT* stands for tri-training

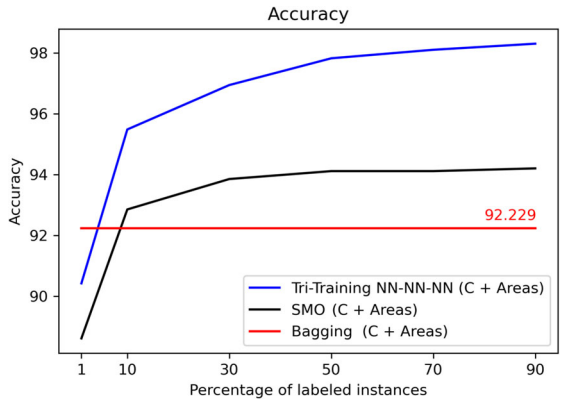


**Fig. 10** Accuracy of different trained models as a function of the percentage of labeled instances in a dataset containing the features CoatingID and the torque signal areas A1–A7

responding linear regression values. The results in Table 8 improved on the results of the SL reference algorithm (SMO), achieving accuracy levels of 94.11% using the 50% labeled dataset and 94.20% using the 90% labeled dataset. The semi-supervised methods achieved accuracy levels of 97.82% using the 50% labeled dataset and 98.30% using the 90% labeled dataset. Those results improved on the results shown in Tables 5, 6, and 7. The SSL methods that obtained the best results were all different combinations of basic learners for the tri-training algorithm (C45C45NN, C45NNC45, NNNNNN).

Comparing the accuracy results in [15], it can be seen that the highest accuracy value among the SL algorithms was obtained by the Bagging algorithm with an accuracy of 92.229% obtained using the dataset containing only the CoatingID and the torque signal areas A1–A7. All other algorithms showed lower accuracy.

Figure 10 shows graphs of the main results: the evolution of the accuracy obtained by the TT-NNNNNN algorithm as a function of the percentage of labeled instances, the accuracy of the SMO algorithm for the same datasets, and the result of the best algorithm obtained in [15]. The SSL TT-NNNNNN algorithm performed better than the SL algorithm in all of the tests.

### 4.5 Extended semi-supervised dataset that replaces the A1, A3, A5, and A7 torque sinusoidal signals areas with a linear regression and includes the linear regression slopes in a sliding window of length 11

Once a better strategy had been found, it was followed up to search for the best performance integrating the sliding window proposed in the bibliography [15]. Table 9 shows the results of using the extended semi-supervised dataset, replacing the torque sinusoidal signals areas A1, A3, A5, and A7 by a linear regression computed from their values, and adding the slopes of a linear regression computed using a sliding window of length 11. The first column shows the percentage of labeled instances, the second column shows the results obtained with the SL reference algorithm (SMO), and the following columns show the SSL algorithms that show the best results for each percentage of labeled instances in the dataset.

As can be seen in Table 9, better results were obtained by replacing the sinusoidal signals with their linear regression and calculating the slope of a linear regression in the sliding window of length 11. The results in Table 9 improved upon the results of the SL reference algorithm (SMO), which achieved accuracy levels of 93.91% using the 50% labeled dataset and 94.21% using the 90% labeled dataset. The semi-supervised methods achieved accuracy levels of 98.20% using the 50% labeled dataset and 98.88% using the 90% labeled dataset, which improved upon the results shown in Tables 5, 6, and 7 and slightly improved upon the results shown in Table 8. The SSL methods that obtained the best results were the Co-Forest algorithm, and different combinations of basic learners for the tri-training (NBSMOSMO) and co-training (SMOSMONN) algorithms.

**Table 9** Accuracy of the different algorithms in percentages

| Labeled instances | Supervised | Semi-supervised methods | | |
|---|---|---|---|---|
| | SMO | TT-NBSMOSMO | CT-SMOSMONN | COFOREST |
| 1% | 73.30% | **74.37%** | 58.92% | 71.23% |
| 10% | 92.52% | 91.44% | **94.86%** | 94.35% |
| 30% | 93.75% | 92.55% | 96.23% | **96.73%** |
| 50% | 93.91% | 92.70% | 96.93% | **98.20%** |
| 70% | 94.06% | 92.85% | 97.14% | **98.65%** |
| 90% | 94.21% | 92.92% | 97.42% | **98.88%** |

Each row represents the results, using a different percentage of labeled instances, ranging from 1 to 90%. The first column contains the dataset percentage of labeled instances. The second column contains the results of the reference SL algorithm (SMO). The remaining columns are the different SSL algorithms that gave the best result for a given dataset. The best result for each row is shown in bold. In the table, $CT$ stands for co-training and $TT$ for tri-training

**Table 10** Time spent training and testing each algorithm per fold and dataset, in seconds

| Labeled instances | Supervised SMO | Semi-supervised methods | | |
|---|---|---|---|---|
| | | TT-NBSMOSMO | CT-SMOSMONN | COFOREST |
| 1% | 1–2 s | 3–5 s | 12–14 s | 1–2 s |
| 10% | 1–2 s | 4–5 s | 8–10 s | 1 s |
| 30% | 1–2 s | 6–8 s | 8–9 s | 1 s |
| 50% | 2 s | 8–11 s | 12–13 s | 1 s |
| 70% | 3 s | 15–20 s | 23–25 s | 1 s |
| 90% | 3–5 s | 24–29 s | 32–37 s | 1–2 s |

Table 10 shows how much time was spent training and testing each fold for the algorithms in Table 9. Training and testing were performed using a machine with two AMD EPYC 7513 processors, each with 32 cores. The machine had 2 TB of RAM memory and no GPU. The programs were executed on an Ubuntu 22.04.1 LTS operating system. As can be seen, some algorithms were faster than others. Although the SMO algorithm was relatively quick to train, as can be seen in the supervised reference column, the training time of the co-training algorithm significantly increased when the SMO algorithm was included as the base learning algorithm. Normally, the testing phase takes a negligible amount of time (i.e., Co-Forest can run tests in less than a millisecond for each fold).

Figure 11 shows the confusion matrix for the test observations from the 30% labeled dataset (on the left) and the 90% labeled dataset (on the right). As can be seen from the figure(s), the algorithm correctly classified most of the observations in both testing subset(s). As expected, the more labeled observations in the dataset, the better the classification. The model also performed worse with the underrepresented (no-go) class.

The graphs of the main results are shown in Fig. 12 and are summarized in Table 9. The results of the SSL CoForest are shown in blue and the comparative SL algorithm (SMO), in black. It also includes the accuracy results from [15] for the 100% labeled dataset. Using the dataset containing only the CoatingID, the torque signal areas A1–A7, and the S1–S7 slopes of the linear regression with a sliding window of length 11, the accuracy levels were 92.827% when using the Bagging algorithm and 93.408% when using the RB-Boost algorithm. However, in [15], more datasets were used. The higher accuracy value was obtained with the simpler ensemble SL Bagging algorithm with an accuracy level of 96.518%, and a better accuracy level of 97.277% was obtained with the more complex ensemble SL SMOTEBoost algorithm. Both results were obtained using the dataset containing the larger number of features: the CoatingID, the torque signal areas A1–A7, the S1–S7 slopes of the linear regression with a sliding window of length 11, the error of the linear regression, and the TSI value. As can be seen in Fig. 12, the SSL CoForest algorithm outperforms all the SL algorithms when using a dataset with 50% or more of labeled instances.

### 4.5.1 Other metrics

However, accuracy may not be the best metric to use when dealing with unbalanced datasets. It may be necessary to look at other metrics that focus on the accuracy achieved for each class in the dataset, rather than the accuracy achieved in the majority class that might be masking poor performance in the other classes. Other metrics were therefore evaluated, such as recall, precision, and F1 score. Those metrics may be defined

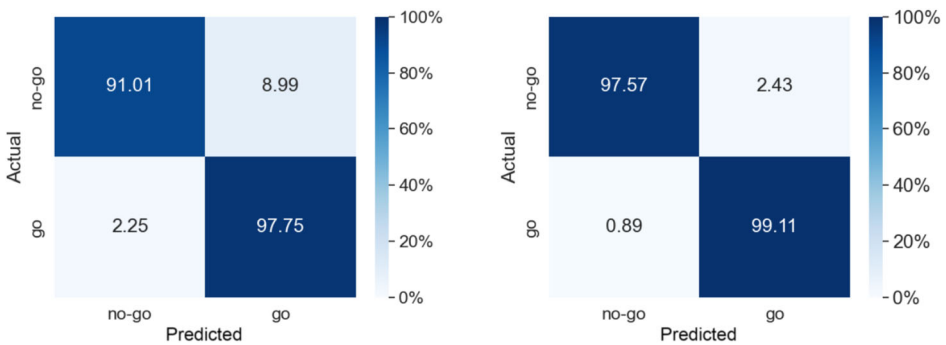


**Fig. 11** Confusion matrix for the Co-Forest algorithm using the test observations from the 30% labeled dataset (left) and 90% labeled dataset (right), in percentages

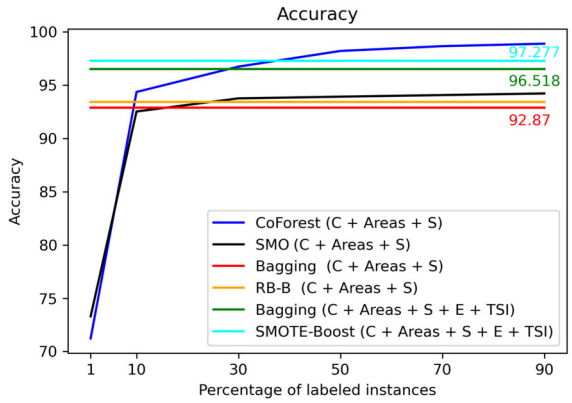


**Fig. 12** Accuracy of the trained models as a function of the percentage of labeled instances within the two datasets. One dataset contained the features CoatingID, torque signal areas A1–A7 and slopes S1–S7. However, two SL methods were also tested on a dataset containing the features: CoatingID, torque signal areas A1–A7 and slopes S1–S7, linear regression error, and TSI. The SL algorithms were trained using a 100% of the labeled instances in the datasets

as follows (using the same criteria found in [15]: the positive class is the one that has fewer instances, and the negative class is the one that has more instances):

$$Recall = TP/(TP + FN) \tag{2}$$

$$Precision = TN/(TN + FP) \tag{3}$$

$$F1\ score = 2 \times (precision * recall)/(precision + recall) \tag{4}$$

Recall, as defined in Eq. 2, is the ratio of true positive (TP) instances divided by the sum of TP and false negative (FN) instances. It refers to the proportion of instances correctly classified as positive out of all instances belonging to the positive class. The higher the recall, the fewer errors occur in classifying the instances of the positive class. Precision, as defined in Eq. 3, is the ratio of true negatives (TN) divided by the sum of TN and false positives (FP). It refers to the proportion of instances correctly classified as negative out of the sum of TN instances plus the negative class instances misclassified as positive. The higher the precision, the fewer the errors there are in classifying negative class instances. The F1 score, as defined in Eq. 4, is considered the harmonic mean of precision and recall.

The above metrics were then evaluated in the last and the most-promising data-treatment strategy, using the same algorithms as shown in Table 9. It can be seen that the recall metric results (Table 11) of the model trained with the SSL Co-Forest algorithm were better than the corresponding results of the other algorithms. Also worth noting is that the model trained with the Co-Forest algorithm obtained a recall metric of 94.38% when trained with the 30% labeled dataset and the recall metric achieved 98.34% when the model was trained with the 90% labeled dataset.

The precision metrics for the trained models in Table 9 are shown in Table 12. As can be seen, the precision metrics of the model trained using the SSL Co-Forest algorithm are, in general, better than the corresponding results of the other models. The model trained with the Co-Forest algorithm obtained a precision metric of 93.07% when trained on the 30% labeled dataset, and the precision metric achieved 97.34% when the model was trained with the 90% labeled dataset.

The F1 metrics of the trained models in Table 9 are shown in Table 13. As can be seen, the F1 metrics of the model trained using the SSL Co-Forest algorithm are, in general, better than the corresponding results of the other models. The model trained with the Co-Forest algorithm obtained a precision metric of 93.72% when trained on the 30% labeled dataset, and the F1 metric scored 97.84% when the model was trained with the 90% labeled dataset.

The results shown in these tree tables (Tables 11, 12, and 13) support the conclusions of the accuracy evaluation of the capabilities of SSL techniques to improve SL techniques applied to datasets with fewer labeled instances, as described

**Table 11** Sensitivity or recall of the different algorithms

| Labeled instances | Supervised SMOSSL | Semi-supervised methods TT-NBSMOSMO | CT-SMOSMONN | COFOREST |
|---|---|---|---|---|
| 1% | 72.68% | 73.00% | 66.47% | **73.56%** |
| 10% | 85.76% | 81.04% | 89.94% | **90.61%** |
| 30% | 87.01% | 82.09% | 92.89% | **94.38%** |
| 50% | 86.94% | 82.61% | 94.26% | **97.27%** |
| 70% | 87.24% | 82.95% | 94.34% | **98.05%** |
| 90% | 87.33% | 82.92% | 94.69% | **98.34%** |

Each row represents the results, using a different percentage of labeled instances, ranging from 1 to 90%. The first column contains the dataset percentage of labeled instances. The second column shows the results of the reference SL SMO algorithm. The remaining columns are the different SSL algorithms that gave the best accuracy metric for a given dataset. The best result for each row is shown in bold. In the table, $CT$ stands for co-training and $TT$ for tri-training

**Table 12** Precision of the different algorithms

| Labeled instances | Supervised | Semi-supervised methods | | |
|---|---|---|---|---|
| | SMOSSL | TT-NBSMOSMO | CT-SMOSMONN | COFOREST |
| 1% | 63.06% | **63.52**% | 58.42% | 62.94% |
| 10% | 85.20% | 84.12% | **89.95**% | 88.13% |
| 30% | 88.18% | 87.25% | 92.45% | **93.07**% |
| 50% | 88.72% | 87.40% | 93.78% | **95.83**% |
| 70% | 89.02% | 87.73% | 94.46% | **96.78**% |
| 90% | 89.48% | 87.99% | 95.18% | **97.34**% |

Each row represents the results, using a different percentage of labeled instances, ranging from 1 to 90%. The first column contains the percentage of labeled instances in the dataset. The second column shows the results of the reference SL SMO algorithm. The remaining columns are the different SSL algorithms that gave the best Accuracy metric for a given dataset. The best result for each row is shown in bold. In the table, $CT$ stands for co-training and $TT$ for tri-training

**Table 13** F1 score of the different algorithms

| Labeled instances | Supervised | Semi-supervised methods | | |
|---|---|---|---|---|
| | SMOSSL | TT-NBSMOSMO | CT-SMOSMONN | COFOREST |
| 1% | 67.53% | **67.93**% | 62.18% | 67.84% |
| 10% | 85.48% | 82.55% | **89.95**% | 89.35% |
| 30% | 87.59% | 84.59% | 92.67% | **93.72**% |
| 50% | 87.82% | 84.94% | 94.02% | **96.55**% |
| 70% | 88.12% | 85.27% | 94.40% | **97.41**% |
| 90% | 88.39% | 85.38% | 94.93% | **97.84**% |

Each row represents the results, using a different percentage of labeled instances, ranging from 1 to 90%. The first column contains the percentage of labeled instances in the dataset. The second column shows the results of the reference SL SMO algorithm. The remaining columns are the different SSL algorithms that yielded the best Accuracy metric for a given dataset. The best result for each row is shown in bold. In the table, $CT$ stands for co-training and $TT$ for tri-training

**Table 14** SFDA, label propagation, and S3VM accuracy metric results

| Labeled instances | Semi-supervised methods | | | |
|---|---|---|---|---|
| | SFDA | Label propagation | Label spreading | S3VM |
| 1% | 0.896232% | 0.849581% | 0.849581% | 0.515969% |
| 10% | 0.912679% | 0.849581% | 0.849581% | 0.509749% |
| 30% | 0.918840% | 0.849581% | 0.849581% | 0.679665% |
| 50% | 0.917464% | 0.850359% | 0.849581% | 0.262919% |
| 70% | 0.914354% | 0.851136% | 0.849581% | 0.673266% |
| 90% | 0.919139% | 0.852931% | 0.849581% | 0.430443% |

Each row represents the results, using a different percentage of labeled instances, ranging from 1 to 90%. The first column contains the percentage of labeled instances in the dataset. The second column shows the results of the reference SFDA algorithm. The following columns show the label propagation and label spreading algorithm results and finally the S3VM algorithm results

in the previous section. Both the recall and precision metrics demonstrate good classification capabilities for the two classes in the different datasets. Since the figures were close and high for each dataset, the F1 score was also high, falling somewhere between the two metrics.

#### 4.5.2 Other SSL approaches

Some alternative SSL approaches were tested to compare them with the methods not included in KEEL. The results are shown in Table 14. The scikit-learn Python package was used to obtain the transductive label propagation results. It contains two algorithms: label propagation and label spreading. Both algorithms were tested. Semi-supervised Fisher Linear Discriminant Analysis (SFDA) was also tested. SFDA was implemented in Python based on the work of [46]. The margin maximization S3VM method [4], was also implemented in Python and tested. It should be noted that label propagation is a transductive approach and that the models must be trained using both the training and the testing subsets, with the testing subset remaining unlabeled. Nevertheless, its results were worse than the results for SFDA, and S3VM obtained the worst results.

It is worth noticing that the results in Table 14 for the dataset with 1% of labeled observations are better than those obtained so far. Other approaches apart from co-training (and its variations) may be capable of obtaining better results in datasets containing very few labeled observations.

## 5 Conclusions

The first test of semi-supervised algorithms to predict tapping process quality in terms of real industrial conditions (binary class and extensive repetitions with different tap tools in terms of tool coatings) has been presented in this study. Data were collected from an internal threading process using taps under high-speed cutting conditions without coolant on nodular cast iron plates. Various supervised and semi-supervised learning methods and algorithms were trained and tested. The models trained using the semi-supervised approach even outperformed those using supervised learning techniques tested on the same original dataset. The main conclusions that can be drawn from this study are as follows:

- First, the SSL approach can achieve the same results as the SL approaches or better. The best accuracy value obtained using the SSL approach was 98.20% using the 50% labeled dataset, yielding higher accuracy levels than the SL SMOTEBoost algorithm that obtained 97.277% as its best result (using a 100% labeled dataset). Therefore, SSL techniques can achieve higher performance (in terms of accuracy, precision, and recall) using half of the data labeled, which is associated with a significant cost reduction.
- Second, the treatment of the features can have a major impact on accuracy. Using the original dataset, which contains sinusoidal fluctuations from experimental perturbations, did not yield the highest performance. However, a simple change in the data representation, such as replacing the sinusoidal signals with a regression with a second degree polinomy, meant that the trained models obtained better results.
- Third, the addition of some additional statistical information, such as the slope of the linear regression with a sliding window of length 11, contributed to better classification capabilities of the trained models.
- Fourth, although the datasets were unbalanced, oversampling techniques (SMOTE) combined with SSL algorithms hardly improved the results.
- Fifth, co-training based algorithms (co-training, tri-training, CoBC, CoForest, …) appeared to mitigate the problems that can arise due to the unbalanced nature of the dataset. Despite the fact that no oversampling or undersampling techniques were used to rebalance the different classes in the dataset, the recall and precision metrics computed were quite good. Surprisingly, all the learning algorithms that achieved better results were co-training-based algorithms.

Future work will be focused on conducting further experiments using semi-supervised techniques in other highly repetitive machining processes. If possible, a more complete study of other SSL algorithms and methods based on alternative approaches, such as label propagation, margin maximization, consistency regularization, and neural networks, should be explored. Also, extensive tuning of the most accurate SSL algorithms might also lead to models of greater accuracy in final industrial implementations, while the modeling of the wear level as a continuous output might also provide interesting academic insight into threading processes. It was surprising that the co-training-based algorithms achieved the best accuracy results, an observation that deserves further attention, as perhaps the use of more than one different base algorithm with a different approach to machine learning can reduce the impact of the unbalanced datasets. Perhaps the presence of a learning algorithm that is less sensitive to unbalanced data could improve the accuracy of the overall classifier. It may also be said that other approaches could reach better results than the co-training approach when using an extremely limited number of labeled observations (in this study case, 76 labeled observations, about 1%).

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1007/s00170-025-16491-x.

**Author contribution** All authors contributed to the study conception and design. Material preparation, data collection, and analysis were performed by Alain Gil-Del-Val. Jose Alberto Maestro-Prieto performed the SL and SSL experiments, and Andrés Bustillo analyzed the results. The first draft of the manuscript was written by Jose Alberto Maestro-Prieto, Alain Gil-Del-Val, and Andrés Bustillo, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

**Funding** Open access funding provided by FEDER European Funds and the Junta de Castilla y León under the Research and Innovation Strategy for Smart Specialization (RIS3) of Castilla y León 2021-2027. This work was supported by the Junta de Castilla y León through project BU055P20 (JCyL/FEDER, UE), the Spanish Ministry of Science and Innovation through project PID2020-119894GBI00/AEI/10.13039/501100011033, and it was co-financed through European Union FEDER funds.
The authors acknowledge Basque Government financing through the ECOVERSO project, the ELKARTEK 2024 program (KK2024/00095), the ORLEGI project, and the ELKARTEK 2024 program (KK2024/00005), and funding from the European Commission through the REINFORCE project, GRANT_NUMBER: 101104204 URL: https://app.dimensions.ai/details/grant/grant.13717357

## Declarations

**Conflict of interest** The authors declare no competing interests.



## References

1. Alloghani M, Al-Jumeily D, Mustafina J, Hussain A, Aljaaf AJ (2020) In: Berry MW, Mohamed A, Yap B.W. (eds.) A systematic review on supervised and unsupervised machine learning algorithms for data science, pp 3–21. Springer. https://doi.org/10.1007/978-3-030-22475-2_1
2. Alshraideh H, Castillo ED, Gil-Del-Val A (2020) Process control via random forest classification of profile signals: an application to a tapping process. J Manuf Process 58:736–748. https://doi.org/10.1016/j.jmapro.2020.08.043
3. Basu S, Banerjee A, Mooney RJ (2002) Semi-supervised clustering by seeding. In: Proceedings of the nineteenth international conference on machine learning. ICML '02, pp 27–34. Morgan Kaufmann Publishers Inc.,
4. Bennett KP, Demiriz A (1998) Semi-supervised support vector machines. In: Proceedings of the 12th international conference on neural information processing systems. NIPS'98, pp 368–374. MIT Press,
5. Bustillo A, Lacalle L, Fernández Valdivielso A, Santos P (2016) Data-mining modeling for the prediction of wear on forming-taps in the threading of steel components. J Comput Design Eng 3(4):337–348. https://doi.org/10.1016/j.jcde.2016.06.002
6. Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: Proceedings of the annual ACM conference on computational learning theory, pp 92–100
7. Breiman L (2001) Random forests. Mach Learn 45(1):5–32. https://doi.org/10.1023/A:1010933404324
8. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357
9. Chen S, He H, Garcia EA (2010) RAMOBoost: ranked minority oversampling in boosting. IEEE T Neural Networ 21(10):1624–1642
10. Chawla NV, Lazarevic A, Hall LO, Bowyer KW (2003) SMOTEBoost: improving prediction of the minority class in boosting. In: Knowledge discovery in databases: PKDD 2003: 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003. Proceedings 7, pp 107–119. Springer
11. Cover T (1968) Estimation by the nearest neighbor rule. IEEE T Inform Theory 14(1):50–55. https://doi.org/10.1109/TIT.1968.1054098
12. Demiriz A, Bennett KP (2001) In: Ferris M.C, Mangasarian O.L, Pang J-S. (eds.) Optimization approaches to semi-supervised learning, pp 121–141. Springer,. https://doi.org/10.1007/978-1-4757-3279-5_6
13. Deng C, Guo MZ (2006) Tri-training and data editing based semi-supervised clustering algorithm. In: Gelbukh A, Reyes-Garcia CA (eds) MICAI 2006: Advances in Artificial Intelligence. Springer, pp 641–651
14. Deng C, Guo MZ (2011) A new co-training-style random forest for computer aided diagnosis. J Intell Inf Syst 36(3):253–281
15. Diez-Pastor JF, Gil-Del-Val A, Veiga F, Bustillo A (2020) High-accuracy classification of thread quality in tapping processes with ensembles of classifiers for imbalanced learning. Measurement 168:108328. https://doi.org/10.1016/j.measurement.2020.108328
16. Díez-Pastor JF, Rodríguez JJ, García-Osorio C, Kuncheva LI (2015) Random balance: ensembles of variable priors classifiers for imbalanced data. Knowl-Based Syst 85:96–111. https://doi.org/10.1016/j.knosys.2015.04.022
17. Fernández-Lucio P, Gil-Del-Val A, Plaza S, Pereira O, Fernández-Valdivielso A, López de Lacalle LN (2023) Threading holder based on axial metal cylinder pins to reduce tap risk during reversion instant. Alex Eng J 66:845–859. https://doi.org/10.1016/j.aej.2022.10.060
18. Gil-Del-Val A, Fernández J, Castillo E, Arizmendi M, Veiga F (2013) Monitoring of thread quality when tapping nodular cast iron with TiN-coated HSS cutting taps. Int J Adv Manuf Technol 69(5):1273–1282. https://doi.org/10.1007/s00170-013-5078-7
19. Gil-Del-Val A, Veiga F, Pereira O, Lacalle LN (2020) Threading performance of different coatings for high speed steel tapping. Coatings 10(5). https://doi.org/10.3390/coatings10050464
20. Gil-Del-Val A, Veiga F, Suárez A, Arizmendi M (2020) Thread quality control in high-speed tapping cycles. J Manuf Mater Process 4(1). https://doi.org/10.3390/jmmp4010009
21. Geßner F, Weigold M, Abele E (2021) Measuring and modelling of process forces during tapping using single tooth analogy process. Prod Eng 15(1):97–107. https://doi.org/10.1007/s11740-020-01004-4

22. Halder A, Ghosh S, Ghosh A (2013) Aggregation pheromone metaphor for semi-supervised classification. Pattern Recogn 46:2239–2248
23. Hady M, Schwenker F, Palm G (2010) Semi-supervised learning for tree-structured ensembles of RBF networks with Co-Training. Neural Netw 23(4):497–509. https://doi.org/10.1016/j.neunet.2009.09.001
24. Huang T, Yu Y, Guo G, Li K (2010) A classification algorithm based on local cluster centers with a few labeled training examples. Knowl Based Syst 26(6):563–571
25. Li M, Zhou Z-H (2005) SETRED: self-training with editing. In: Ho TB, Cheung D, Liu H (eds) Advances in Knowledge Discovery and Data Mining. Springer, pp 611–621
26. Li M, Zhou ZH (2007) Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. Systems and Man and Cybernetics and Part A Systems and Humans and IEEE Transactions on 37(6):1088–1098
27. Li M, Zhou Z-H (2007) Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. IEEE Transactions on Systems Man and Cybernetics - Part A Systems and Humans 37(6):1088–1098. https://doi.org/10.1109/TSMCA.2007.904745
28. Martins PS, Santos JO, Carneiro JRG, Silva GC, Dias CAR, Vieira VF, Fernandes GHN, Ba ECT (2022) Study of wear behavior and tool life in different taps during the internal threading of a nodular iron engine crankshaft. Int J Adv Manuf Technol 120(11):7803–7814. https://doi.org/10.1007/s00170-022-09290-1
29. Monka P, Monkova K, Modrak V, Hric S, Pastucha P (2019) Study of a tap failure at the internal threads machining. Eng Fail Anal 100:25–36. https://doi.org/10.1016/j.engfailanal.2019.02.035
30. Montgomery DC (2020) Introduction to statistical quality control. John Wiley & Sons, Incorporated
31. Ma Y-C, Wan M, Yang Y, Zhang W-H (2019)c Dynamics of tapping process. Int J Mach Tools Manuf 140:34–47. https://doi.org/10.1016/j.ijmachtools.2019.02.002
32. Oezkaya E, Biermann D (2018) Development of a geometrical torque prediction method (GTPM) to automatically determine the relative torque for different tapping tools and diameters. Int J Adv Manuf Technol 97(1):1465–1479. https://doi.org/10.1007/s00170-018-2037-3
33. Pereira I, Bacci Da Silva M (2017) Study of the internal thread process with cut and form taps according to secondary characteristics of the process. Int J Adv Manuf Technol 93. https://doi.org/10.1007/s00170-017-0573-x
34. Pereira IC, Vianello PI, Boing D, Guimarães G, Silva MB (2020) An approach to torque and temperature thread by thread on tapping. The Int J Adv Manuf Technol 106(11):4891–4901. https://doi.org/10.1007/s00170-020-04986-8
35. Ramírez-Sanz JM, Maestro-Prieto J-A, Arnaiz-González BA (2023) Semi-supervised learning for industrial fault detection and diagnosis: A systemic review. ISA Trans 143:255–270. https://doi.org/10.1016/j.isatra.2023.09.027
36. Seiffert C, Khoshgoftaar TM, Van Hulse J, Napolitano A (2009) RUSBoost: A hybrid approach to alleviating class imbalance. IEEE transactions on systems man and cybernetics-part A systems and humans 40(1):185–197
37. Toher D, Downey G, Murphy TB (2011) Semi-supervised linear discriminant analysis J Chemom 25(12):621–630. https://doi.org/10.1002/cem.1408
38. Triguero I, González S, Moyano JM, García S, Alcalá-Fdez J, Luengo J, Fernández A, Jesús MJ, Sánchez L, Herrera F (2017) KEEL 3.0: an open source software for multi-stage analysis in data mining. Int J Comput Intell Syst 10:1238–1249. https://doi.org/10.2991/ijcis.10.1.82
39. Tao X, Ren C, Li Q, Guo W, Liu R, He Q, Zou J (2021) Bearing defect diagnosis based on semi-supervised kernel Local Fisher Discriminant Analysis using pseudo labels. ISA Trans 110:394–412. https://doi.org/10.1016/j.isatra.2020.10.033
40. Engelen JE, Hoos HH (2020) A survey on semi-supervised learning. Mach Learn 109(2):373–440. https://doi.org/10.1007/s10994-019-05855-6
41. Wang J, Luo S-w, Zeng X-h (2008) A random subspace method for co-training. In: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), pp 195–200. https://doi.org/10.1109/IJCNN.2008.4633789
42. Yarowsky D (1995) Unsupervised word sense disambiguation rivaling supervised methods. In: 33rd Annual meeting of the association for computational liguistics, pp 189–196
43. Yaslan Y, Cataltepe Z (2010) Co-training with relevant random subspaces. Neurocomputing 73(10):1652–1661. https://doi.org/10.1016/j.neucom.2010.01.018
44. Zhou Y, Goldman S (2004) Democratic co-learning. In: 16th IEEE international conference on tools with artificial intelligence, pp 594–602. https://doi.org/10.1109/ICTAI.2004.48
45. Zhou ZH, Li M (2005) Tri-training: exploiting unlabeled data using three classifiers. IEEE Trans Knowl Data Eng 17:1529–1541
46. Zheng J, Wang H, Song Z, Ge Z (2019) Ensemble semi-supervised Fisher discriminant analysis model for fault classification in industrial processes. ISA Trans 92:109–117. https://doi.org/10.1016/j.isatra.2019.02.021