APEM journal

Advances in Production Engineering & Management
Volume 20 | Number 2 | June 2025 | pp 277–290
https://doi.org/10.14743/apem2025.2.540

ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

A transfer learning approach to machine learning-based end-of-line quality inspection

Mlinarič, J.[a,b,*], **Pregelj, B.**[a], **Boškoski, P.**[a], **Petrovčič, J.**[a], **Dolanc, G.**[a]

[a]Jozef Stefan Institute, Ljubljana, Slovenia
[b]Jozef Stefan International Postgraduate School, Ljubljana, Slovenia

ABSTRACT

Transfer learning is a powerful machine learning technique for accelerating the learning process of a new classification task by using knowledge from an existing and related classification task that has already been trained. This technique addresses situations where the necessary training and test data are unavailable or where data acquisition is costly, difficult, or impractical. Additionally, transfer learning significantly reduces training time compared to training from scratch. This study demonstrates the benefits of transfer learning in developing an end-of-line quality inspection system to produce brushless DC electric motors during preproduction. Decision Tree, Random Forest, Bagging, and AdaBoost classifiers were trained on a dataset of 10,000 instances from a mass-produced motor subtype (A). Knowledge in the form of hyperparameters and feature importance was transferred to classifiers for two preproduction subtypes (B and C), each with only 100 instances. The results show up to a 20 % improvement in classification accuracy and significantly lower misclassification costs when using transfer learning. The study highlights the importance of transfer learning as an effective approach for improving industrial classification tasks, especially in preproduction phases where datasets are typically small and imbalanced.

ARTICLE INFO

Keywords:
Quality inspection;
Fault detection;
Machine learning;
Transfer learning;
Decision trees;
Random forests;
Bagging;
AdaBoost

**Corresponding author:*
jernej.mlinaric@ijs.si
(Mlinarič, J.)

Article history:
Received 12 February 2025
Revised 23 June 2025
Accepted 27 June 2025

1. Introduction

End-of-line (EoL) quality inspection contributes significantly to the manufacturing processes and to the overall quality of released products [1, 2]. With proper quality inspection, faulty products can be detected, diagnosed, and separated from good ones. Quality inspection can isolate and recognize faults of products; therefore, those products can either be repaired or recycled [1].

To establish quality inspection, extensive specific knowledge is required [2, 3]. This knowledge is usually possessed by experts, who gained it from past experiences and studies [1, 4]. With the advancement of artificial intelligence (AI), quality inspection systems can be established using machine learning (ML) methods [5]. These methods can significantly reduce dependence on expert knowledge. However, they require a vast amount of learning, testing and validating data.

The problem arises when the production of new product subtype starts, and there is not enough historical learning data, since the data has yet to be generated and collected during the production. Therefore, the ML process cannot start immediately, but only after the sufficient

amount of data is collected. The learning data must include instances of all expected classes, including rare errors, which may occur, for example, once in every 1,000 instances. Therefore, during the preproduction and test production stage (e.g., 100 instances of products), a sufficient amount of data is often not available.

However, when planning to manufacture the new product on existing production lines where structurally similar products have already been produced in large quantities, some knowledge from existing quality inspection algorithms can be utilized for the inspection of new product type in limited quantities. This procedure is called Transfer Learning (TL).

The study presented in this paper is based on real industrial data, obtained from an EoL quality inspection system of manufacturing line, installed at one of the European mass producers of electric motors. On the presented line, various motor types are produced, but this paper focuses on a particular motor class (referred to as 720). The research investigates whether the knowledge gained from motor classifiers for the mass-produced subtype can be transferred and applied to the ML of classifiers for similar preproduction motor subtypes, which have limited data quantities due to the early preproduction stage, where only a few (up to 100) units are produced.

This paper is organized as follows: Section 2 describes the problem, subject of inspection and existing quality inspection system. Section 3 discusses the transfer learning, studied methods and procedure. Section 4 presents the analysis of results, and finally, Section 5 provides the conclusion.

2. Problem description

The addressed EoL quality inspection system is part of the fully automated production lines at Domel, Slovenia, a renowned mass producer of electric motors. In this section, the subject of quality inspection and quality inspection systems are briefly described.

2.1 Subject of quality inspection

The subject of inspection is the brushless DC (BLDC) motor class named ‘720’, and shown in Fig. 1. It is designed for battery-driven applications, such as electric garden tools, handheld tools, service robots, small appliances, and automotive uses. It is produced in many variants, defined by supply voltage (12-48 V), stack height (10-40 mm), nominal speed (3,000-45,000 rpm), power range (200-2,000 W), and the presence and type of control electronics [6].

This paper examines three subtypes of the motor class 720, referred to as A, B, and C. Subtype A has already been mass-produced, while the other two (B and C) were in preproduction stages, resulting in limited production quantities. The observed motor subtypes are produced at the same manufacturing line, and they have very similar structures but differ in only a few parameters (such as size, power, capacity, voltage, etc.).

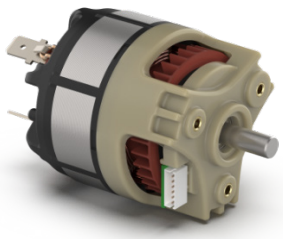

Fig. 1 BLDC motor class 720 (subject of quality inspection)

2.2 Existing EoL quality inspection system

Production line is equipped with a modular End-of-Line (EoL) quality inspection system (Fig. 2) that inspects each produced motor. Each motor undergoes several short test runs to measure various parameters (electrical properties, rotation speed, vibrations, and sound at different speeds of rotation). In addition to motor parameters, the auxiliary variables are measured and used to compensate the effect of environment (air temperature, pressure and humidity). The measurements produce time-series waveforms, or "raw signals," sampled at 10-60 kHz for 0.1 to 1 second, resulting in several time series of different sizes (from 1,000 up to 30,000 samples).

To reduce data volume and, most importantly, to extract the relevant information, raw signals are processed by digital filtering, down-sampling, averaging, and finally, frequency analysis. Details about this have already been documented in [7] and in [8-10]. The result of signal processing is a set of features, where each feature is represented by a single floating-point scalar value. Typically, feature represents signal power at a particular frequency. In case of motor class 720 and selected subtypes, the quality inspection algorithm operates with 80 features, originating from sound, vibration, and electric signals. Based on the values of the observed features and their specified ranges, the inspected motors are categorized into three categories: GOOD, BAD and UNDEFINED. These categories are detailed in Table 1 and explained in [7]. Range of each feature (upper and lower threshold) is usually determined by experts, who also select the subset of features most relevant for the quality inspection.

Table 1 Diagnostic result generation [7]

Measurement status	Features	Diagnostic result
Completed	All features are within specified ranges	GOOD
	One or more features are outside specified range	BAD
Not completed, due to: • measurement faults • sensor faults • motor manipulation and transport faults etc.		UNDEFINED

Fig. 2 Modular EoL quality inspection system

Table 2 Key steps in the development of inspection system.

Step	Description	Executor
1. Selecting inspected parameters	Define inspection process, focusing on electrical, mechanical and performance tests.	Production engineers and experts
2. Designing the inspection process	Develop the workflow, including test sequences, manipulation procedures and database structure.	Production engineers and experts
3. Signal processing	Select and integrate appropriate sensors, capture real-time analog signals and convert them into digital data (details in [7-9] and [11])	Production engineers and experts
4. Data processing	Design algorithm to extract features from data to identify specific product characteristics. This process provides actual measurement value (details in [7-9, 11]).	Production engineers and experts
5. Classification task	Design algorithms to categorize inspected items into categories: GOOD, BAD, UNDEFINED. This step involves: 1. *Feature selection;* identify relevant features for inspection. 2. *Threshold adjustment and tuning;* setting and fine-tuning threshold values.	AI, ML (presented in [7])
6. Classifier adaptation	Adjust the inspection system to accommodate new product types or subtypes.	AI and ML (this study)
7. Data archive	Design system to store results in the company database, monitor manufacturing processes and detect trends such as system wear, degradation or material changes.	Production engineers and experts

The development of an EoL quality inspection system is a complex, challenging, and time-consuming process. Building such a system requires experts with specialized knowledge, and as a result, the system heavily relys on human expertise. To minimize reliance on experts, the ML approach was used to support *automated feature selection* and *automated feature threshold adjustment* [7]. The development process of the presented system follows the structured steps, outlined in Table 2.

3. Transfer learning for quality inspection

Transfer Learning (TL) is a ML technique where knowledge gained from training a model on one task (the source domain) is applied to a different but related task (the target domain) [12]. In this study, the idea of TL is to transfer knowledge from previously established quality inspection algorithms of mass-produced motor subtype (the source domain) to the new preproduction motor subtype that only has limited training data (the target domain). This approach can achieve the following goals:

- Develop a reliable quality inspection algorithm using a reduced amount of training data,
- Reduce ramp-up time and expedite the development of the quality inspection process.

3.1 TL basics

TL is effectively utilized in a wide range of applications like computer vision, especially for image classification tasks [13], natural language processing (e.g. BERT, GPT, and ELMo [14-16]), speech recognition [17], medical imaging [18], time-series forecasting [19] and robotics [20].

TL is usually applied with artificial neural networks and deep learning [12, 21] and it is less frequently used with ML classifiers such as Decision Trees and Ensembles (Random Forests, Bagging and AdaBoost). However, several studies [22-25] demonstrate that TL can also be applied to these classifiers in the following ways:

1. *Instance transfer:* In this approach, the suggested classifiers are trained on the source domain data and then applied to the target domain [24, 26]. The idea is that some data from source domain (instances from mass-production) are reweighted (their impact on the new model is reduced) and then directly reused for training new models with limited data (instances from preproduction). The success of transfer learning increases with the similarity between source and target domain. When instant transfer is performed, classifiers of source domain and target domain do not necessarily share any similarities, but they share some training data, as shown in Fig. 3, section a) in Target domain rectangle.
2. *Feature transfer:* This approach focuses on finding a list of relevant features of source domain and transferring them to the target domain [27, 28]. In the case of Decision Trees and Ensemble classifiers, this can be implemented by analysing the feature importance in the source domain. The assumption is that the feature importance in the target domain will closely resemble that of the source domain. When feature transfer is performed, the training data for classifiers of the source and target domain and the structure of classifiers remain independent. However, both classifiers share the same features, although their threshold values may differ (Fig. 3, section b in Target domain rectangle).
3. *Parameter transfer:* In this approach, the parameters of classifier are transferred from source to target domain [29]. In this case, instead of training from scratch, the initial splits and structures of source domain classifiers can be fine-tuned for purposes of target domain classification. In models like Decision trees and Ensembles, parameters such as weights, feature thresholds, estimators, and leaf node prediction as well as hyperparameters like splitting criteria, maximal depth, number of estimators, can be transferred. Parameters are values, learned during ML processes, while hyperparameters are pre-set values that control the structure and behaviour of the model [30]. When parameter transfer is performed, the training data of source and target domain remain independent, but the structure of the classifier (or part of it) can be the same. Please see Fig. 3, section c) in Target domain rectangle.

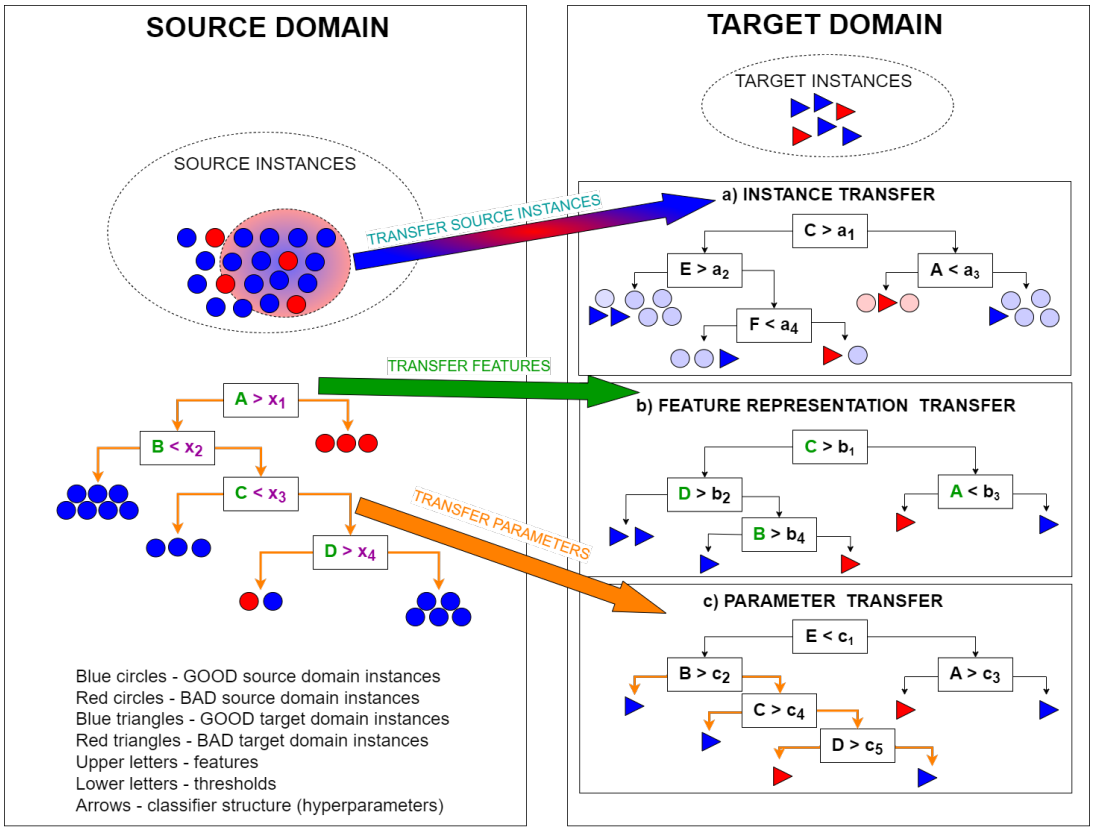

Fig. 3 Explained TL: a) Instance transfer, where source instances are added and mixed to target instances to ensure a sufficient amount of data (light colors represents lower weight), b) feature transfer, where all features (upper case letters) from source domain are transferred and used for target domain classification, however, they do not necessarily be on the same place, indicating different classifier structure, c) parameter transfer, where parameters that define classifier structure are transferred to target domain classifier (orange arrows represent part of structure that is defined by hyperparameters), one can notice that source and target structure of the decision tree are the same, but features at each node can be different.

In our case of EoL Quality Inspection system, *Decision Trees*, *Random Forests*, and *Bagging* and *AdaBoost* classifiers were applied to implement *Feature transfer* and *Parameter transfer*. For feature transfer, the most important features—determined through feature selection and feature importance metrics—were identified in the source domain (mass-produced motor subtype A) and transferred to the ML classifiers used in the target domain (preproduction subtypes B and C). For parameter transfer, hyperparameters were transferred. *Instance transfer* was not employed in this study due to several key limitations:

- The source and target domains are structurally similar, but not identical, which can lead to distributional shifts in feature behaviour and fault manifestation. Direct reuse or re-weighting of source-domain instances in the target domain may introduce label noise or reduce classifier reliability.
- The dataset size imbalance is substantial. Subtype A includes 10,000 instances, whereas subtypes B and C contain only 100 each. This would likely bias the classifier toward the source domain, undermining generalization.
- The signal pre-processing already condenses raw sensor data into abstracted, denoised features, reducing the marginal value of reusing raw instances.
- Due to communication infrastructure limitations in the production environment, large-scale data manipulation is practically infeasible for the considered application.

While both feature and parameter transfer enhanced performance in this study, some trade-offs exist. By focusing on the most informative features, feature transfer enabled the creation of simplified, generalizable models, especially in cases where training data is limited. However, by eliminating less important features carries the risk of overlooking subtle fault indicators. By

reusing optimized hyperparameters, parameter transfer allows faster convergence. However, it may be less robust to changes in feature importance or domain shifts. The combined use of both methods helped mitigate these limitations and contributed to the development of reliable and robust classifiers for preproduction stage.

3.2 Chosen ML classifiers

To utilize TL, supervised ML classifiers were chosen due to the availability of labelled data. These classifiers provide several benefits, such as statistical reliability, robustness, and a decreased reliance on expert knowledge (e.g., understanding the physical background of the system). However, implementing supervised ML classifiers require a substantial amount of data from the production process. These classifiers are well-suited for manufacturing lines engaged in mass production. In this study, the following classifiers were used and compared:

- *Decision Tree classifier (DT)* involves recursive partitioning of the instance space into a tree structure. The top nodes, or roots, have no incoming edges, whereas internal nodes, or test nodes, divide the space based on attribute values. These internal nodes represent decision points and at the bottom ones-leaves-indicate decision outcomes [31, 32].
- *Random Forest classifier (RF)* is one of the most powerful ensemble learning techniques, combining several tree predictors. It belongs to a class of averaging methods where several independent estimators are built and their predictions averaged [33, 34].
- *Bagging (bootstrap aggregating) classifier (BG)* is a basic method used in creating an ensemble of classifiers. Similarly to averaging methods, bagging combines outputs of a number of classifiers with random training sets; therefore, improving accuracy [34, 35].
- *AdaBoost (adaptive boosting) classifier (AB)* is a widely used ensemble classifier that combines multiple weak classifiers to form a stronger classifier. It involves the repeated training of weak classifiers such that each iteration focuses on the instances previously misclassified, thereby improving model accuracy. In classification tasks, AdaBoost is quite effective at reducing bias and variance [34, 36].

The optimal structure of classifiers was determined through hyperparameter tuning, while feature selection and feature importance analysis generated a ranked list of relevant features, ordered from the most to the least important for classification. The feature selection process had already been discussed in [7] and is not the subject of this study.

Parameters are the internal variables that are automatically learned during the training process and directly control the model's predictions. Hyperparameters are the settings or configurations of the model that are set before the training and determine the structure of the model. In this study, hyperparameters are being tuned via hyperparameter optimizers to ensure the best classifier structure. Since SciKit learn python library was utilized, the following optimizers were available:

- Grid Search Cross Validation (CV) [37],
- Halving Grid Search CV [38],
- Halving Random Search CV [39],
- Parameter Grid [40],
- Parameter Sampler [37],
- Randomized Search CV [41].

In this work, hyperparameters were optimized using *Halving Random Search CV method* (with 10-fold Cross-Validation), which combines Halving Grid Search and Randomized Search. The process begins with a random set of hyperparameter combinations, after which the algorithm identifies the optimal combination. This method balances exploration and exploitation and is more efficient than traditional Grid Search [39].

Hyperparameters to be tuned for each classifier are listed in Table 3. Due to computational complexity, only the most influential hyperparameters (according to [31-35], [30]) are being tuned.

Table 3 List of transferable hyperparameters for each classifier.

DT	RF	BG	AB
Criterion	Bootstrap	Base estimator	Base estimator
Max depth	Criterion	Bootstrap	Learning rate
Max features	Max depth	Max features	N estimators
Min samples split	Max features	Max samples	
Min samples leaf	Min samples split	N estimators	
Splitter	Min samples leaf		
	N estimators		

Base estimator refers to the learning algorithm used to construct the ensemble in ensemble classifier such as Bagging and AdaBoost [36].

Bootstrap determines whether the entire dataset or a randomly sampled subset is used to build each tree in a Random Forest [42].

Criterion defines the function that measures the quality of a split in decision trees, with GINI impurity being a common choice for classification tasks [31].

Learning rate adjusts the contribution of each base learner within the ensemble, influencing how quickly the model adapts during training [43].

Max depth limits the maximum depth of the trees to prevent overfitting and complexity [31].

Max features sets the maximum number of features considered when splitting a node, when expressed as a float, it indicates a fraction of the total features [33].

Max samples specifies the fraction of samples drawn from the original dataset to train each base estimator [42].

Min samples leaf defines the minimum number of samples required to be present in a leaf node [31].

Min samples split defines the minimum number of samples necessary to split an internal node [31].

N estimators present the number of base estimators in the ensemble [33].

Splitter specifies the strategy employed to split each node during tree construction [33].

3.3 TL data

For machine learning, measured data of motor subtypes A, B, and C and were available. As mentioned, subtype A was mass-produced, so larger dataset was available. Each dataset (A, B and C) was divided into two parts: training data (75 % of all data) and testing data (25 % of all data). Instances for training and testing were selected randomly. The same training and testing subsets were used for all ML classifiers. A summary of the observed subtypes is provided in the Table 4.

In the table, 80 represents the number of features in the datasets, while 10,000 and 100 correspond to the number of inspected motors and the sizes of the respective datasets. The number 1 in the output array indicates a single output variable, i.e., the diagnostic result (GOOD/BAD). Each feature represents a pre-processed value, obtained from raw signals, measured by sensors during EoL testing. Noise and faults present during the manufacturing process are accounted for during signal processing, which includes filtering, down-sampling, normalization and feature extraction procedures. As a result, the datasets reflect meaningful, denoised inputs suitable for robust machine learning classification.

Table 4 Available data of observed motor subtypes.

Subtype	Input matrix	Output array	Production stage
A	80×10000	1×10000	Mass production
B	80×100	1×100	Preproduction
C	80×100	1×100	Preproduction

3.4 TL implementation

Fig. 4 outlines the complete procedure, which unfolds as follows:

1. *Data acquisition for mass-produced motor subtype.* The feature set of mass-produced motor subtype is obtained by data acquisition and signal processing, detailed in [7-9] and [11].
2. *Training classifiers.* The collected data of motor subtype A is used for training the classifiers. During training, best hyperparameters are determined using Halving Random Search, as explained in Subsection 3.2.

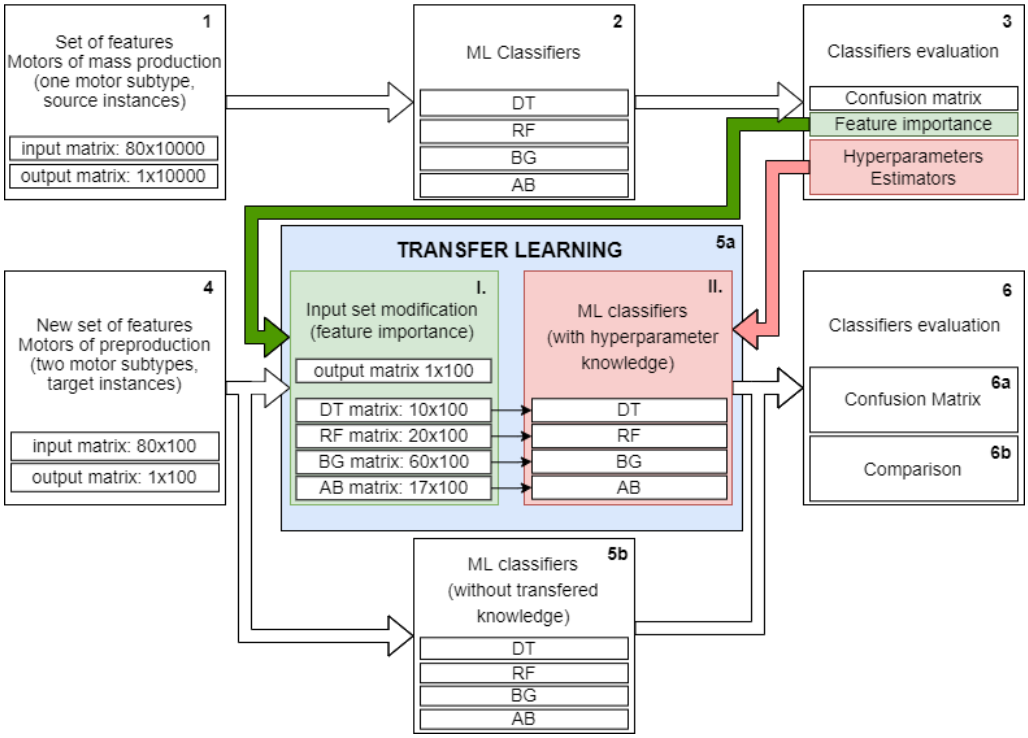

Fig. 4 Flow chart of the procedure

3. *Classifier evaluation*. Trained classifiers are evaluated across three main criteria:
   a) Confusion Matrix (CM) analysis addresses accuracy and performance [7].
   b) Evaluation of the importance of particular features of input data [7, 44]. The knowledge gained from this step is later utilized as feature transfer.
   c) Extraction of hyperparameters and estimators, and evaluation for new classifiers of preproduction subtypes. The knowledge obtained in this step is transferred through parameter transfer.

4. *Data acquisition for preproduction motor subtypes*. Like the procedure in step 1, the data of preproduction motor subtypes are collected and features extracted.

5. *Divergence into two training workflows*. At this point, procedure splits into two ways:
   a) *With transfer learning*. This procedure is executed in two steps:
      I. *Input set reduction (feature transfer).* The dataset from step 4 is reduced based on feature importance evaluation for each classifier from step 3. This reduced dataset is then used in step 5a II. The size of new reduced input matrix for each classifier is shown in Fig. 4, rectangle 5a I.
      II. *Training with reduced dataset and transferred hyperparameters (parameter transfer).* The optimal hyperparameters and estimators from step 3 are transferred to the new classifiers, which are then trained using reduced dataset from step 5a I. Each classifier utilized its own reduced dataset and corresponding transferred hyperparameters and estimators.
   b) *Without transfer learning*. Input data from step 4 is directly used for training classifiers, follows step 6. This involves creating classifiers without any transferred knowledge, solely for comparison purposes. Training procedure follows the step 2.

6. *Evaluation and comparison*. This procedure split into two steps:
   a) *Classifier evaluation through CM*. The performance (accuracy and miss-classification cost) of classifiers from step 5a and 5b are evaluated
   b) *Classifier comparison*. Classifiers from step 5a and 5b are evaluated and compared through accuracy and mis-classification cost value.

Note: Steps 4 to 6 must be repeated twice (for motor subtypes B and C). Also, note that classifier performance was evaluated using accuracy and cost value, where higher accuracy (closer to 1 is better) and lower misclassification cost value (closer to 0 is better) indicate a better classifier [7].

However, when dealing with imbalanced data, where typically only 3-10 % of motor instances (depending on the motor subtype) represent BAD motors, accuracy can be misleading. A classifier may achieve high accuracy simply by classifying all instances as GOOD. Therefore, CM and cost-sensitive learning have been employed, and the performance of classifiers is heavily influenced by the misclassification cost value [7].

4. Analysis of the results

4.1 Training the classifiers of mass-produced subtype (motor subtype A, source domain)

The classifier for the mass-produced motor subtype A was developed by steps 1-3 of Fig. 4. Its performance was evaluated based on accuracy and cost value, as shown in Table 5, rows *Accuracy* and *Miss-classification cost value*.

Next, the feature importance was assessed, and the insights were used to modify the input set for a new set of features (Step 5a I, Fig. 4). The number of important features influences the complexity of classifier. Classifier with fewer important features is simpler, but selecting too few features could result in undetected faults. Conversely, a high number of important features can result in an overly complex classifier. The number of important features transferred to the classifiers of preproduced subtypes is listed in Table 5, row *Number of important features.*

In Table 5, rows *Base estimator* to *Splitter* shows the optimal hyperparameter values for each trained classifier for motor subtype A. These values are the results of hyperparameter tuning process since they have been adjusted automatically. These optimized settings were then transferred to the classifiers for motor subtypes B and C (Step 5a II, Fig. 4).

Table 5 Specifications of classifiers for motor subtype A

	DT	RF	BG	AB
Accuracy	0.996	0.9956	0.9956	0.9932
Miss-classification cost value	91	92	101	152
Number of important features	10	20	60	17
Base estimator			Bagging	AdaBoost
Bootstrap		False	False	
Criterion	Gini	Gini		
Learning rate				1,2
Max depth	None	50		
Max features	0.8	0.5	0.8	
Max samples			0.8	
Min samples leaf	1	2		
Min samples split	2	10		
N estimators		200	200	50
Splitter	Best			

4.2 Training the classifiers of preproduced subtypes (B and C)

The classifiers of preproduced subtypes B and C were trained, evaluated, and compared according to 3.4 (steps 4 to 6). The results of the comparison are presented in Table 6.

The data in Table 6 demonstrate that classifiers with transferred knowledge (column ''With TL'') consistently outperform those without transferred knowledge (column ''Without TL''), achieving higher accuracy (with three classifiers reaching 100 % accuracy) and significantly lower miss-classification cost. In contrast, classifiers without TL, due to high miss-classification cost are unsuitable for industrial purposes. The beneficial impact of TL is particularly evident for subtype C, where the DT improved accuracy from 76-96 % and miss-classification cost dropped from 15 to 1.

All evaluated classifiers with TL were found as effective and well-suited for industrial classification tasks. Among them, the DT classifier with the lowest miss-classification cost, stands out as the most appropriate. The BG and RF classifiers also prove as effective, offering slightly higher accuracy but also higher miss-classification cost for subtype C. On the other hand, the AB classifier with the highest miss-classification cost is somewhat less suitable. The performance of each individual classifier for each motor subtype is presented as CM in Figs. 5-20.

Table 6 Performance comparison of the classifiers for preproduction subtypes

Motor Subtype	Classifier method	Without TL		With TL	
		Accuracy	Miss-classification cost	Accuracy	Miss-classification cost
B	DT	0.92	2	0.96	1
	RF	0.92	2	1	0
	BG	0.96	10	1	0
	AB	0.96	10	1	0
C	DT	0.76	15	0.96	1
	RF	0.84	13	0.96	10
	BG	0.8	50	0.96	10
	AB	0.84	40	0.92	20

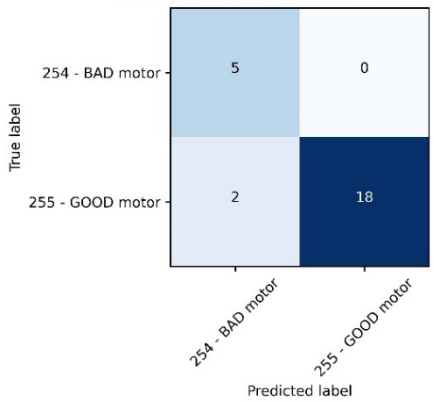

Fig. 5 CM for DT classifier without TL, subtype B

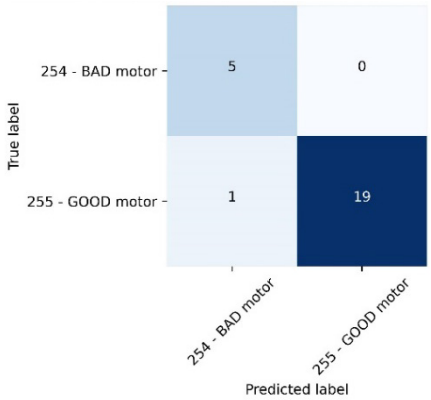

Fig. 6 CM for DT classifier with TL, subtype B

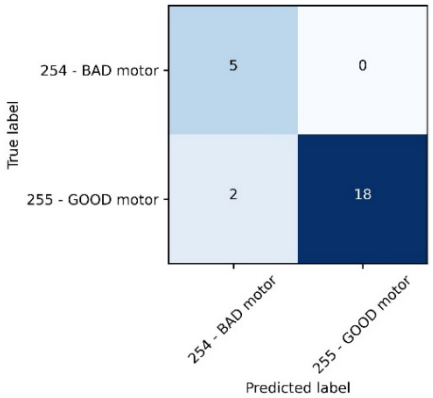

Fig. 7 CM for RF classifier without TL, subtype B

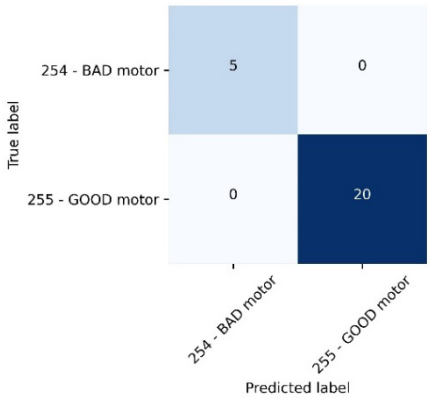

Fig. 8 CM for RF classifier with TL, subtype B

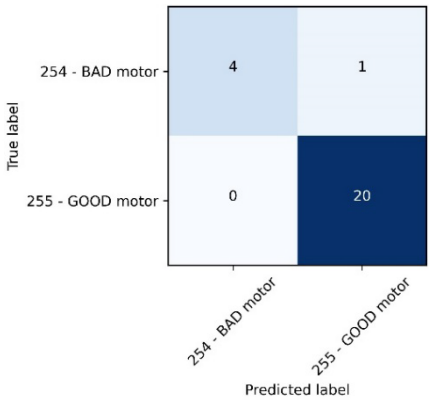

Fig. 9 CM for BG classifier without TL, subtype B

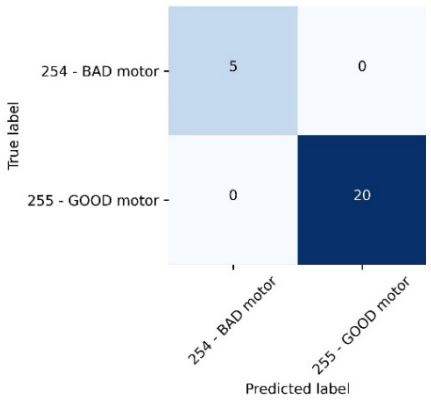

Fig. 10 CM for BG classifier with TL, subtype B

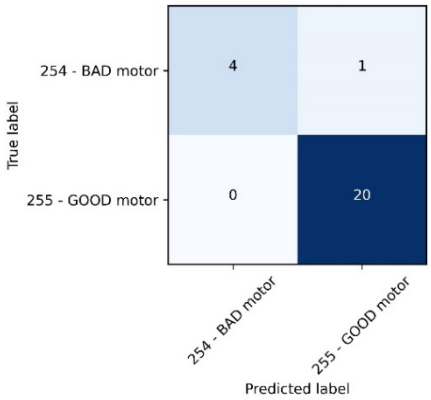

Fig. 11 CM for AB classifier without TL, subtype B

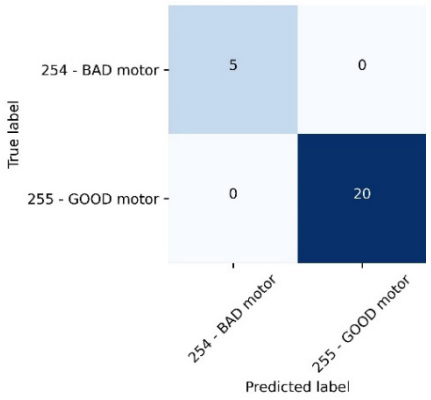

Fig. 12 CM for AB classifier with TL, subtype B

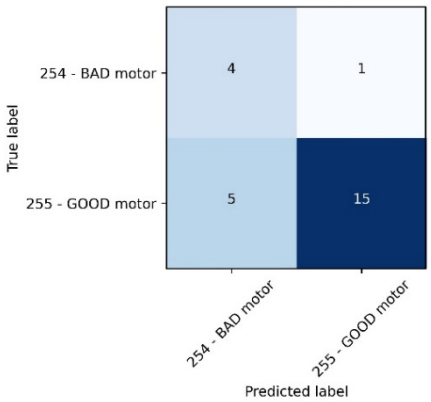

Fig. 13 CM for DT classifier without TL, subtype C

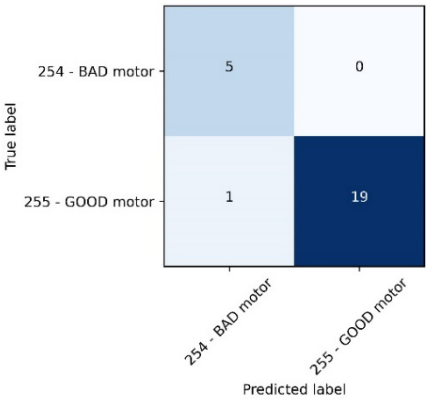

Fig. 14 CM for DT classifier with TL, subtype C

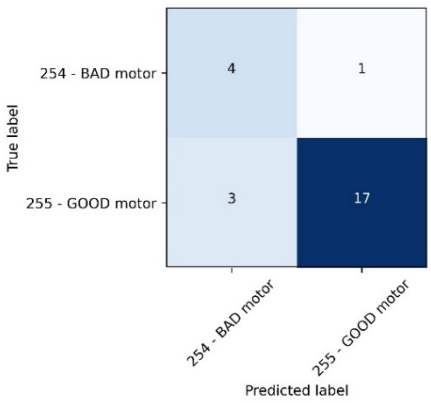

Fig. 15 CM for RF classifier without TL, subtype C

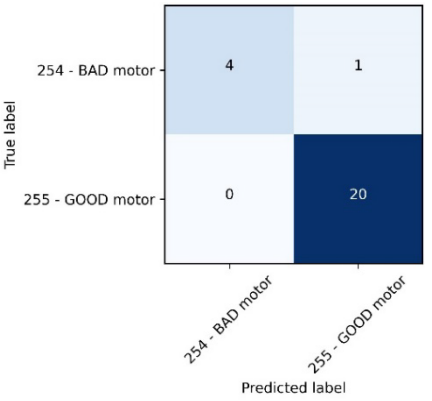

Fig. 16 CM for RF classifier with TL, subtype C

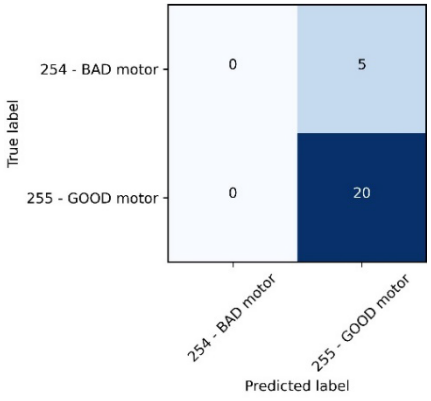

Fig. 17 CM for BG classifier without TL, subtype C

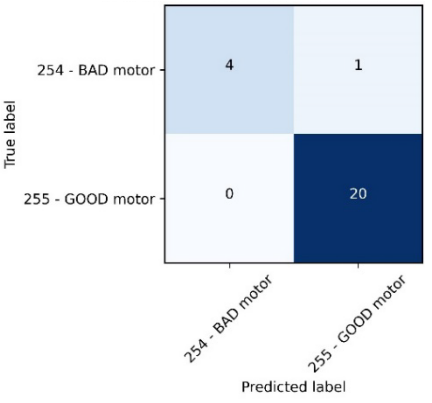

Fig. 18 CM for BG classifier with TL, subtype C

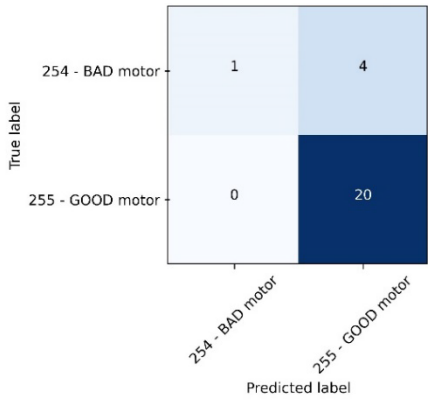

Fig. 19 CM for AB classifier without TL, subtype C

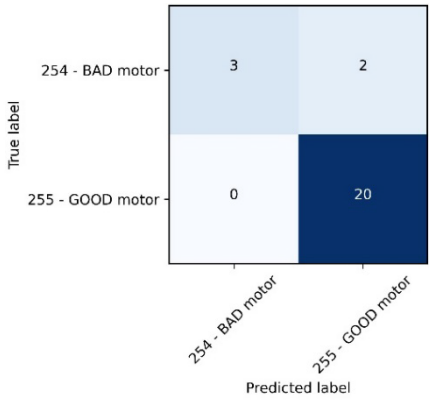

Fig. 20 CM for AB classifier with TL, subtype C

5. Conclusion

In this study, Decision Tree, Random Forest, Bagging, and AdaBoost classifiers were used for EoL quality inspection in the mass production of electric motors. The primary goal is to classify each produced motor as Good or Bad based on acquired electric parameters, vibration and sound during a short test run. The mentioned four classifiers were initially trained on a large dataset of 10,000 instances originating from the motor subtype, which is mass-produced. The knowledge gained during training was then transferred to the new classifiers for the new preproduced motor subtypes, where available training data were limited to only 100 instances per subtype.

The results clearly demonstrate the substantial advantages of employing TL in commissioning automated EoL quality inspection systems for electric motors. By leveraging TL, reliable classifiers for quality inspection systems were developed for new motor subtypes using significantly reduced datasets. This enables the establishment of quality inspection systems earlier in the preproduction phase, prior to the accumulation of extensive data. As additional data for new motor subtypes become available, the EoL inspection system can be continuously refined and improved.

In addition to improving model accuracy and enabling early deployment, the proposed approach has minimal impact on inference time and production latency. Classifiers operate on preprocessed features extracted during existing signal processing routines, and the tree-based models used are computationally lightweight; therefore, classification can be performed within milliseconds. This ensures real-time or near-real-time decision-making without disrupting the production flow.

This study focused on three structurally similar motor subtypes from the same production line. However, the approach is expected to generalize well to other motor types produced within the company. Most production lines at Domel share similar manufacturing processes, diagnostic procedures, and feature structures. Therefore, the TL framework presented here, holds the possibility for broader applicability across different motor types and even other production lines, provided appropriate pre-processing and feature selection are performed.

The TL-based classifiers are integrated into a larger diagnostic pipeline that includes safety measures and fail-safes to reduce the risk of misclassifying critical detects. Generally, motors that are marked as "UNDEFINED" or that are close to decision thresholds are marked for manual examination or further testing. Additionally, redundant sensing channels (such as vibration, sound and electrical signals) are employed in the inspection process to improve the reliability of fault detection. The impact of possible misclassifications is lessened by these built-in safeguards, especially in high-risk fault cases.

Overall, the findings confirm TL as a robust and efficient technique for improving classification performance in industrial applications, particularly in situations involving imbalanced or limited datasets.

Acknowledgments

This work was supported by the Slovenian Research Agency under Grants P2-0001 and L2-4454.

References

[1] Ngadiman, Y., Hussin, B., Talib Bon, A., Abdul Hamid, N.A. (2017). Factors that influenced the quality inspection on the production line in manufacturing industry, In: *Proceedings of 2016 the 3rd International Conference on Mechatronics and Mechanical Engineering,* Shanghai, China, Article No. 10007, doi: 10.1051/matecconf/20179510007.
[2] Borkowski, S., Knop, K. (2016). Challenges faced in modern quality inspection, *Management and Production Engineering Review*, Vol. 7, No. 3, 11-22, doi: 10.1515/mper-2016-0022.
[3] Hinckley, C.M. (1997). Defining the best quality-control systems by design and inspection, *Clinical Chemistry*, Vol. 43, No. 5, 873-879, doi: 10.1093/clinchem/43.5.873.
[4] Womack, J., Jones, D., Roos, D. (1990). *The machine that changed the world: The story of lean production, Toyota's secret weapon in the global car wars that is now revolutionizing world industry*, Free Press, New York, USA.
[5] Sundaram, S., Zeid, A. (2023). Artificial intelligence-based smart quality inspection for manufacturing, *Micromachines*, Vol. 14, No. 3, Article No. 570, doi: 10.3390/mi14030570.
[6] Domel d.o.o.. Domel 720 inner runner motor, from *https://www.domel.com/product/720-inner-runner-motor-24*, accessed June 19, 2024.
[7] Mlinarič, J., Pregelj, B., Boškoski, P., Dolanc, G., Petrovčič, J. (2024). Optimization of reliability and speed of the end-of-line quality inspection of electric motors using machine learning, *Advances in Production Engineering & Management*, Vol. 19, No. 2, 182-196, doi: 10.14743/apem2024.2.500.
[8] Benko, U., Petrovčič, J., Juričić, D. (2005). In-depth fault diagnosis of small universal motors based on acoustic analysis, *IFAC Proceedings Volumes*, Vol. 38, No. 1, 323-328, doi: 10.3182/20050703-6-CZ-1902.01856.
[9] Benko, U., Petrovčič, J., Mussiza, B., Juričić, D. (2008). A system for automated final quality assessment in the manufacturing of vacuum cleaner motors, *IFAC Proceedings Volumes*, Vol. 41, No. 2, 7399-7404, doi: 10.3182/20080706-5-KR-1001.01251.
[10] Boškoski, P., Petrovčič, J., Musizza, B., Juričić, D. (2011). An end-quality assessment system for electronically commutated motors based on evidential reasoning, *Expert Systems with Applications*, Vol. 38, No. 11, 13816-13826, doi: 10.1016/j.eswa.2011.04.185.
[11] Juričić, D., Petrovčič, J., Benko, U., Musizza, B., Dolanc, G., Boškoski, P., Petelin, D. (2013). End-quality control in the manufacturing of electrical motors, In: Strmčnik, S., Juričić, Đ. (eds.), *Case studies in control. Advances in industrial control,* Springer, London, UK, 221-256, doi: 10.1007/978-1-4471-5176-0_8.
[12] Weiss, K., Khoshgoftaar, T.M., Wang, D. (2016). A survey of transfer learning,, *Journal of Big Data,* Vol. 3, No. 1, Article No. 9, doi: 10.1186/s40537-016-0043-6.
[13] Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S. (2014). CNN features off-the-shelf: An astounding baseline for recognition, In: *Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops,* Columbus, USA, 512-519, doi: 10.1109/CVPRW.2014.131.
[14] Devlin, J., Chang, M.W., Lee, K., Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding, In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1, Minneapolis, USA, 4171-4186.
[15] Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., Tang, J. (2023). GPT understands, too, *ArXiv*, doi: 10.48550/arXiv.2103.10385.
[16] Affi, M., Latiri, C. (2021). BE-BLC: BERT-ELMO-based deep neural network architecture for English named entity recognition task, *Procedia Computer Science*, Vol. 192, 168-181, doi: 10.1016/j.procs.2021.08.018.
[17] Watanabe, S., Hori, T., Hershey, J.R. (2017). Language independent end-to-end architecture for joint language identification and speech recognition, In: *Proceedings of 2017 IEEE Automatic Speech Recognition and Understanding Workshop,* Okinawa, Japan, 265-271, doi: 10.1109/ASRU.2017.8268945.
[18] Tajbakhsh, N., Shin, J.Y., Gurudu, S.R., Hurst, R.T., Kendall, C.B., Gotway, M.B., Liang, J. (2016). Convolutional neural networks for medical image analysis: Full training or fine tuning?, *IEEE Transactions on Medical Imaging*, Vol. 35, No. 5, 1299-1312, doi: 10.1109/TMI.2016.2535302.
[19] Thill, M., Konen, W., Wang, H., Bäck, T. (2021). Temporal convolutional autoencoder for unsupervised anomaly detection in time series, *Applied Soft Computing*, Vol. 112, Article No. 107751, doi: 10.1016/j.asoc.2021.107751.
[20] Finn, C., Abbeel, P., Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks, In: *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, 1126-1135.
[21] Pan, S.J., Yang, Q. (2010). A survey on transfer learning, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 22, No. 10, 1345-1359, doi: 10.1109/TKDE.2009.191.
[22] Ben-David, S., Blitzer, K., Crammer, K., Pereira, F. (2007). Analysis of representations for domain adaptation, In: Schölkopf, B., Platt, J.,Hofmann, T. (eds.), *Advances in neural information processing systems 19: Proceedings of the 2006 conference,* MIT Press, Cambridge, Massachusetts, USA, 137-144, doi: 10.7551/mitpress/7503.003.0022.
[23] Vu,T.V., Shi, Z., Cheng, J., Zhang, Q., He, K., Wang, S., Harrison, R.M. (2019). Assessing the impact of clean air action on air quality trends in Beijing using a machine learning technique, *Atmospheric Chemistry and Physics*, Vol. 19, No. 17, 11303-11314, doi: 10.5194/acp-19-11303-2019.
[24] Dai, W., Yang, Q., Xue, G.-R., Yu, Y. (2007). Boosting for transfer learning, In: *Proceedings of ICML '07 & ILP '07: The 24th Annual International Conference on Machine Learning held in conjunction with the 2007 International*

Conference on Inductive Logic Programming, Corvallis, USA, 193-200, doi: 10.1145/1273496.1273521.
[25] Pan, S.J., Yang, Q. (2010). A survey on transfer learning, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 22, No. 10, 1345-1359, doi: 10.1109/TKDE.2009.191.
[26] Jiang, J., Zhai, C. (2007). Instance weighting for domain adaptation in NLP, In: Zaenen, A., van den Bosch, A. (eds.), *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Prague, Czech Republic, 264-271.
[27] Lee, S.-I., Chatalbashev, V., Vickrey, D., Koller, D. (2007). Learning a meta-level prior for feature relevance from multiple related tasks, In: *Proceedings of the ICML '07 & ILP '07: The 24th Annual International Conference on Machine Learning held in conjunction with the 2007 International Conference on Inductive Logic Programming,* Corvalis, USA, 489-496, doi: 10.1145/1273496.1273558.
[28] Argyriou, A., Micchelli, C.A., Pontil, M., Ying, Y. (2006). A spectral regularization framework for multi-task structure learning, In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*, Vancouver, Canada, 1-8.
[29] Acharya, A., Hruschka, E.R., Ghosh, J., Acharyya, S. (2012). Transfer learning with cluster ensembles, In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, Bellevue, Washington, USA, 123-132.
[30] Hastie, T., Tibshirani, R., Friedman, J. (2009). *The elements of statistical learning*, Springer, New York, USA, doi: 10.1007/978-0-387-84858-7.
[31] Breiman, L., Friedman, J., Olshen, R.A., Stone, C.J. (1984). *Classification and regression trees*, Chapman and Hall/CRC., New York, USA, doi: 10.1201/9781315139470.
[32] Rokach, L., Maimon, O. (2014). *Data mining with decision trees*, 2nd edition, World Scientific, New Jersey, USA, doi: 10.1142/9097.
[33] Breiman, L. (2001). Random forest, *Machine Learning*, Vol. 45, No. 1, 5-32, doi: 10.1023/A:1010933404324.
[34] Rokach, L. (2010). *Pattern classification using ensemble methods*, World Scientific, Singapore, doi: 10.1142/9789814271073.
[35] Breiman, L. (1996). Bagging predictors, *Machine Learning*, Vol. 24, No. 2, 123-140, doi: 10.1023/A:1018054314350.
[36] Freund, Y., Schapire, R.E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences*, Vol. 55, No. 1, 119-139, doi: 10.1006/jcss.1997.1504.
[37] Bergstra, J., Bengio, Y. (2012). Random search for hyper-parameter optimization, *Journal of Machine Learning Research*, Vol. 13, 281-305.
[38] Jamieson, K., Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization, In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, Cadiz, Spain, 240-248.
[39] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization, *Journal of Machine Learning Research*, Vol. 18, 1-52.
[40] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. (2011). Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research*, Vol. 12, 2825-2830.
[41] Bergstra, J., Bardenet, R., Bengio, Y., Kegl, B. (2011). Algorithms for hyper-parameter optimization, In: *Proceedings of the 24th International Conference on Neural Information Processing Systems,* Granada, Spain, 2546-2554.
[42] Efron, B., Tibshirani, R.J. (1994). *An introduction to the bootstrap*, CRC Press, New York, USA, doi: 10.1201/9780429246593.
[43] Friedman, J.H. (2001). Greedy function approximation: A gradient boosting machine, *Annals of Statistics*, Vol. 29, No. 5, 1189-1232, doi: 10.1214/aos/1013203451.
[44] Deng, X., Li, Y., Weng, J., Zhang, J. (2019). Feature selection for text classification: A review, *Multimedia Tools and Applications,* Vol. 78, No. 3, 3797-3816, doi: 10.1007/s11042-018-6083-5.
[45] Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep learning*, MIT Press, Cambridge, USA.