Constructing a meta-learner for unsupervised anomaly detection

MAŁGORZATA GUTOWSKA[1], SUZANNE LITTLE[1], AND ANDREW MCCARREN[1]

[1]School of Computing, Dublin City University, Dublin, Ireland

Corresponding author: Małgorzata Gutowska (e-mail: malgorzata.gutowska2@mail.dcu.ie).

ABSTRACT Unsupervised anomaly detection (AD) is critical for a wide range of practical applications, from network security to health and medical tools. Due to the diversity of problems, no single algorithm has been found to be superior for all AD tasks. Choosing an algorithm, otherwise known as the *Algorithm Selection Problem* (ASP), has been extensively examined in supervised classification problems, through the use of meta-learning and AutoML, however, it has received little attention in unsupervised AD tasks. This research proposes a new meta-learning approach that identifies an appropriate unsupervised AD algorithm given a set of meta-features generated from the unlabelled input dataset. The performance of the proposed meta-learner is superior to the current state of the art solution. In addition, a mixed model statistical analysis has been conducted to examine the impact of the meta-learner components: the meta-model, meta-features, and the base set of AD algorithms, on the overall performance of the meta-learner. The analysis was conducted using more than 10,000 datasets, which is significantly larger than previous studies. Results indicate that a relatively small number of meta-features can be used to identify an appropriate AD algorithm, but the choice of a meta-model in the meta-learner has a considerable impact.

INDEX TERMS Anomaly detection, unsupervised anomaly detection, algorithm selection problem, model selection, meta-learning, meta-features.

I. INTRODUCTION

ANOMALY detection (AD) is gaining increased attention in various business sectors, primarily due to the growing number of systems that collect and use data generated through a variety of daily activities. Detecting anomalies in sophisticated systems not only helps them run smoothly, but in many domains, an identified anomaly is of genuine value. An outlier can be a potentially harmful event that should be avoided or an early indicator of a new trend. Insurance, finance, network traffic monitoring, and health are just a few examples of the many different fields that benefit from AD applications [1]–[4].

In contrast to typical classification tasks, most real-world anomaly detection problems are unsupervised. Anomalies are rarely known in advance since they have not yet been identified, and those that could have been detected may differ significantly from those still to be discovered. This is intuitive given the essential properties of anomalies, as they are markedly different from other data instances and rare compared to normal data points [5].

To overcome the challenges posed by the difficult to define character of anomalies, researchers categorise them in a variety of ways, such as by their relationship to other data points [6]:

- point anomaly – an individual data instance deviating from other data, often referred to as global anomaly [5], [7], [8],
- collective anomaly – a collection of data instances anomalous with respect to the entire dataset,
- contextual anomaly – a data instance anomalous only in a specific context but not otherwise, also referred to as a local anomaly [8].

Typically, AD algorithms seek to estimate the likelihood of every data point being an anomaly in an unsupervised manner [5], considering the anomaly characteristics and types mentioned above. These algorithms return an anomaly score (usually not normalised), where a higher score indicates an increased likelihood that a given data point is an anomaly [5], [6]. It is not possible, therefore, to measure an algorithm's performance in the form of a single metric such as accuracy,
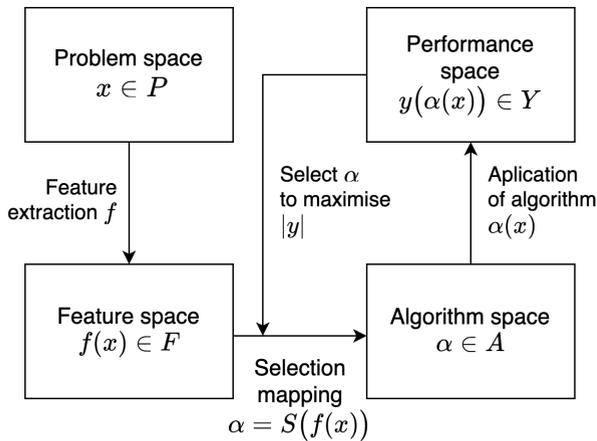
FIGURE 1: Rice's framework, as presented by Smith-Miles [15].

recall or precision. A detailed study of the results is usually necessary to assess the performance for a new task.

A. ALGORITHM SELECTION PROBLEM

Although numerous AD techniques have been proposed, it is generally accepted that no single method is optimal for all AD problems [1], [2]. The concern of a single algorithm performing well on a limited number of problems is not unique to AD. Rather, it is common across machine learning tasks, and is otherwise known as the *No Free Lunch Theorem* [9]. The problem of selecting the best algorithm for a given task is referred to as *Algorithm Selection Problem* (ASP) [10]–[12]. Conventional approaches to the ASP are not always optimal. For example, the trial-and-error strategy can easily become time-consuming, and hiring a human expert in a specific field can be prohibitively expensive. Furthermore, none of these strategies guarantees success.

An alternative method to overcome these drawbacks is the automated selection of the best performing algorithm. Creating a model that learns from historical evaluations is a well-established approach to ASP, also known as meta-learning [13], [14]. Meta-learning, along with Automated Machine Learning (AutoML), of which meta-learning is an essential part, is a rapidly developing topic in machine learning [14].

Among the most popular strategies for meta-learning is the use of dataset characteristics that can be mapped to the performance of specific algorithm configurations from historical evaluations [14], [15]. Rice's seminal paper [16] conceptualised the problem of algorithm selection by mapping problem characteristics to algorithm performance. This framework was expanded by Smith-Miles [15] and captured in the form shown in Fig. 1.

There are four essential components of the framework [15]:

- problem space P – a set of dataset instances of a problem,

- feature space F – a set of characteristics generated from each dataset instance x,
- algorithm space A – a set of algorithms (possibly including variations incorporating hyperparameters),
- performance space Y – a set of performance metrics of the algorithms from A evaluated over the problem space P.

The ASP can be defined as "For a given problem instance $x \in P$ with features $f(x) \in F$, find the selection mapping $S(f(x))$ into algorithm space A, such that the selected algorithm $\alpha \in A$ maximises the performance mapping $y(\alpha(x)) \in Y$" [15].

The ASP is even more challenging when faced with an unsupervised task. Applying a trial-and-error approach or maximisation of performance mapping is problematic without a single performance metric. A meta-learner that indicates the best algorithm for a new unsupervised task is thus a necessary tool, particularly in scenarios with high up-front uncertainty and high variability.

Few studies have proposed an automated algorithm selection in AD [17]–[21]. Of these, only one model is designed for unsupervised scenarios and utilises the meta-learner framework as presented by Rice [18].

The current work has been motivated by the growing importance of AD tools and the little research done to address the problem of algorithm selection in unsupervised AD scenarios. In this research, an alternative meta-learner for unsupervised AD has been developed with respect to its three components: meta-feature space F, AD algorithm space A, and the meta-model (i.e., a model that learns the mapping of the meta-features into the algorithm space [22], $S(f(x))$, referred to as m in the following part of this manuscript). A comparison with the state of the art and a statistical analysis of eight meta-learner variants (three factors, each with two levels) was then performed.

This study makes several contributions to the algorithm selection problem in unsupervised anomaly detection. The main ones include:

- creation of an alternative unsupervised AD meta-learner, superior to the existing state of the art solution,
- examining the effect size of the components contributing to the performance of the meta-learner.

In addition, this study presents secondary contributions:

- proposal of an extension of Rice's framework, which is refactored to the unsupervised AD problem,
- implementation of an existing state of the art unsupervised meta-learning technique on an AD benchmark project consisting of more than 10,000 diverse datasets.

The remainder of the paper is structured as follows: Section II provides an overview of work in the meta-learning and AutoML domains, as well as work in the algorithm selection problem for unsupervised anomaly detection. The conducted experiments are described in Section III, and the results are reported and discussed in Sections IV and V, respectively. The study's conclusions are presented in Section VI.

Since the formalisation of the ASP framework by Rice, meta-learning has received increased attention in the research community. Studies have focused on specific areas of meta-learning, such as hyperparameter optimisation (HO) [23]–[30] or meta-features development [31]–[33]. Several survey studies have also presented a large-scale overview of the research [12], [13], [15], [34], [35]. In particular, a comprehensive compilation of the research and achievements in meta-learning and broader fields such as AutoML is presented in [14]. Current AutoML systems are not only capable of performing model or hyperparameter selection tasks, but are also fully functional automated pipelines of processes that include training and testing without the need for human intervention [36]. To date, several automated AutoML systems have been developed, including Auto-WEKA [37], Auto-Sklearn [38], Auto-Sklearn 2.0 [39], Hyperopt-Sklearn [23], Auto-Net [40] and others [41], [42].

These systems were developed to solve supervised problems. The ASP for unsupervised settings remains mostly unexplored. A few studies have examined the meta-features' potential in AD scenarios [1], [18], [19], [43], but their limitations are either due to the required knowledge of data labels or a structure, or a need for an extensive evaluation of algorithms for a new task. The approach presented in [18] has addressed the ASP problem for unsupervised tasks by utilising the knowledge gained through past algorithm evaluation and is the state of the art solution for the aforementioned problem.

A key early exploration of this challenge at scale by Campos et al. [1] examined how well the selected AD methods handled anomalies across datasets. The authors defined two dataset properties, which were difficulty and diversity. The difficulty score indicated, for each AD method, the difficulty of detecting outliers in a given dataset. The diversity score reflected the agreement between the methods on the difficulty score. The researchers created a feature map based on these two scores and positioned the datasets accordingly. This study explored meta-analysis of the algorithm's performance across datasets, and is viewed as a first step toward inferring dataset meta-features and developing a system to aid in algorithm selection.

Kandanaarachchi et al. [43] attempted to create a method for automated algorithm selection in AD problems by analysing the problem space (referred to as instance space). Two factors that could impact algorithm performance were investigated. These were dataset normalisation and dataset characteristics (meta-features). Two meta-features derived from an initial set of 362 features were used to characterise the datasets and create a two-dimensional space filled with the dataset instances. The feature space outlined optimal regions for a specific method and normalisation type. This study examined 12,000 datasets obtained by downsampling and repurposing classification datasets, yielding the most extensive published set of AD benchmark datasets to date. However, the meta-features developed in this work required

labelled data to be present, making the chosen features unsuitable for an unsupervised AD problem.

An alternative set of meta-features for AD tasks has been proposed by Kotlar et al. [19]. Their research focused on semantic features designed explicitly for AD, such as global, local, or collective anomaly types, as well as other characteristics like anomaly space, anomaly ratio, data type, or data domain. Creating these features requires prior knowledge of anomalous instances, such as their ratio or distribution. These features also need to be provided by a human expert rather than extracted automatically from the datasets. This work cannot, therefore, be directly applied to an automated unsupervised meta-learning problem.

Recent work from Zhao et al. [18] has provided a system (referred to as MetaOD or UOMS, Unsupervised Outlier Model Selection) for predicting the best algorithm for an unsupervised AD task. This work utilised inherent dataset characteristics that do not require labelled data and are generated automatically from datasets. In addition to the statistical meta-features common to many classification tasks, landmarking features specific to AD problems were also included. A collection of algorithms are assessed covering classic AD algorithms, each combined with a set of hyperparameters to yield 302 potential AD models. The experiments were conducted on two test setups with 100 and 63 datasets, respectively. The meta-model was specifically designed to address the algorithm selection problem in AD and is based on collaborative filtering (CF), employing the matrix factorisation method. This approach is further highlighted in Section III, and, as a state of the art solution, it forms a key part of the comparative analysis in the current study.

A different approach of addressing the ASP in AD to that mentioned above has recently been presented in [20], [21]. These studies consider the unsupervised nature of AD problems but do not use offline meta-training. Instead, the systems for assessing the performance of a predefined set of algorithms "online", while evaluating them on a new task, were proposed. The primary shortcoming of these systems is non-scalability. They require evaluation (or even multiple evaluations) of a number of algorithms when confronted with a new problem, without taking advantage of transfer learning from historically evaluated algorithms.

Although ASP and AutoML have received a lot of interest overall, limited research has examined the ASP in unsupervised AD, i.e. when labels for a task at hand are unknown. To date, the most successful solution presented in the research is the UOMS outlined by [18]. The lack of practical benchmarks or potential directions for the improvement of future systems is another obstacle. In most cases, the trend in ASP for AD problems tends to focus on the development of meta-features and not the meta-model or the range of applicable algorithms that can be chosen by the meta-learner. Understanding the influencing factors in a meta-learner is key for the future development of meta-learning systems. In addition, little attention has been given to creating an extensive dataset that can truly test a proposed meta-learner.

This section outlines an alternative strategy to that proposed by Zhao et al. [18], which uses fewer meta-features and has a neural network (NN) as opposed to CF as the base learner. An extensive dataset is required to validate this approach, and the process for generating this data is initially proposed. An evaluation process is then outlined, which attempts to identify the components that contribute to the identification of the best performing algorithm when using meta-learning for a range of unsupervised AD algorithms.

A. DATA GENERATION

The problem/dataset space P for this experiment was built on one of the largest available unsupervised AD dataset repositories. A total of $N = 10,000$ datasets were randomly chosen from those proposed in [43]. Each dataset is defined as x_i where $i = 1, \ldots, N$.

To generate the performance space Y, a selection of AD algorithms $\alpha_j \in A$ were evaluated over all the datasets in P. During the algorithm implementation, two performance metrics were captured: the area under the receiver operating characteristic curve (AUC) and the average precision (AP).

The AUC and the AP are the most widely used evaluation measures in anomaly detection [1], [2]. The output of an AD method is typically an anomaly score, which indicates the likelihood that the observation is anomalous. The translation of anomaly scores into binary labels that are required in common performance metrics, such as accuracy or precision, involves using a threshold, which in many practical AD cases is unknown without additional data exploration. The AUC and AP measures provide the integral result for all thresholds from 0 to 1, making them threshold-independent.

The result of this experiment stage were two matrices of the performance values $Y_\beta \in R^{N \times L}$, where $\beta \in \{AUC, AP\}$, whereas N and L denote the number of the datasets and AD algorithms, respectively.

Procedure 1 outlines the steps taken for generating the data of the performance values Y_{AUC} and Y_{AP} for the AD algorithm set A and the set of datasets P.

The details of the AD algorithms used are discussed in Section III-B. The algorithm performance metrics obtained at this stage are available at http://ieee-dataport.org/10491.

B. PROPOSED META-LEARNER

The meta-learner proposed in the current study has three component parts: meta-features (F_1), a set of base AD algorithms (A_1), and the base learner – meta-model (m_b). Each component is outlined below.

1) Meta-features

The current study uses a significantly smaller set of meta-features (F_1) compared to UOMS [18] (19 vs 200), which were specifically designed to accommodate a broad range of

---

Procedure 1 AD algorithms performance data generation

Input:
$$A = \{\alpha_j \mid j = 1, \ldots, L\}$$
$$P = \{x_i \mid i = 1, \ldots, N\}$$

Output:
$$Y_{AUC} \in R^{N \times L}$$
$$Y_{AP} \in R^{N \times L}$$

1: **for** $j = 1$ to L **do**
2:     **for** $i = 1$ to N **do**
         {Run algorithm α_j over dataset x_i}
3:         $\begin{aligned} y_{AUCij} \\ y_{APij} \end{aligned} \; \alpha_j(x_i)$
4:     **end for**
5: **end for**
6: **return** Y_{AUC}, Y_{AP}

---

anomaly characteristics. The overall design criteria for the features are as follows:

- to be independent of the data labels (suitable for unsupervised scenarios),
- to capture the main types of anomalies, such as global, local, and collective,
- to describe multivariate characteristics of the data (mutual relations between the data points and their neighbourhood).

There are two steps involved in the process of feature generation: 1) calculation of distances between data points; 2) obtaining the actual features as statistical characteristics of the acquired distance distributions.

In step 1, an approach inspired by Moran scatterplots depicting the relationship between global and local z-scores [7] has been proposed to express characteristics related to local and global anomalies. The approach has been generalised to multivariate datasets by substituting the Mahalanobis distance for z-scores. Global and local Mahalanobis distances, δ_G^i and δ_{Ln}^i, were calculated for each data point i as expressed in (1) and (2):

$$\delta_G^i(z_i) = \sqrt{(z_i - \mu)^T S^{-1}(z_i - \mu)} \tag{1}$$

$$\delta_{Ls}^i(z_i) = \sqrt{(z_i - \mu_s)^T S_s^{-1}(z_i - \mu_s)} \tag{2}$$

where $z_i = (z_{1i}, \ldots, z_{Ki})$ represents a single observation (a data point) with K features, $\mu = (\mu_1, \mu_2, \ldots, \mu_K)$ represents the mean of all other observations in the dataset, and S is the covariance matrix. In (2), μ_s and S_s represent the mean and covariance matrix of the s nearest neighbours of z_i, and δ_{Ls}^i defines the Mahalanobis distance to the respective s nearest neighbours. The number of nearest neighbours has been chosen using a grid search approach and defined as $s \in \{20, 60, 80\}$. After this step, each data point i has been represented by four distances to the rest of the data:

$$\delta_G^i; \; \delta_{L20}^i; \; \delta_{L60}^i; \; \delta_{L80}^i.$$

TABLE 1: Meta-features proposed in the F_1 set.

Meta-feature	Instances	Count
TR		4
CM		4
TH	$: \{$ $_G$, $_{L20}$, $_{L60}$, $_{L80}\}$	4
TQ		4
L_s	s: {20, 60, 80}	3

In step 2, the statistical characteristics of each of the distance profiles have been obtained. For each set of $_t$ and $_G$ the features such as TotalRange TR, CenterMass CM, TailHalf TH, and TailQuarter TQ have been calculated as expressed in (3)–(6):

$$TR = \max(^i) - \min(^i) \tag{3}$$

$$CM = \frac{1}{TR}\left[P^{75}(^i) - P^{25}(^i)\right] \tag{4}$$

$$TH = \frac{1}{TR}\left[\max(^i) - P^{50}(^i)\right] \tag{5}$$

$$TQ = \frac{1}{TR}\left[\max(^i) - P^{75}(^i)\right] \tag{6}$$

where is one of $_G$, $_{L20}$, $_{L60}$, $_{L80}$ and P^{25}, P^{75}, and P^{50} are the 25th and 75th percentiles and the median, respectively. In addition, for each neighbourhood $_s$ a property aiming to describe a dataset "locality" L_s was calculated using (7):

$$L_s = \frac{1}{N}\sum_i^N \frac{^i_{Ls}}{^i_G} \tag{7}$$

with N representing the total number of data instances within the dataset. The resulting features are summarised in Table 1.

2) Set of anomaly detection algorithms

The requirement to select methods from a few diverse families, in particular, several methods from the "classic bucket", and modern deep learning-based techniques, drove the selection for the base set of meta-learner algorithms. The A_1 set includes ten "classic" and three neural network-based AD algorithms. Table 2 presents the algorithms and their parameters chosen for A_1. In contrast to UOMS [18], this collection did not focus on examining the algorithm hyperparameters. This is therefore referred to as "w/o HO". The lack of the HO makes this set more distinct from UOMS that is employed for the comparative evaluation.

The conventional methods involved in the experiments cover the following AD algorithms: Local Outlier Factor (LOF) [44], k-Nearest Neighbours (kNN) [45], One-Class Support Vector Machines (OCSVM) [46], Connectivity-Based Outlier Factor (COF) [47], Angle-based Outlier Detector (ABOD) [48], Isolation Forest (iForest) [49], Histogram-based Outlier Score (HBOS) [50], Lightweight Online Detector of Anomalies (LODA) [51], Copula-based Outlier Detection (COPOD) [52], and Principal Component Analysis-based anomaly detection (PCA) [53]. Deep learning-based approaches include Variational Autoencoder (VAE) [54],

TABLE 2: AD models and their parameters comprising the set proposed in this study (A_1).

AD algorithm	Parameter 1	Parameter 2
LOF	n_neighbors = 60	distance = 'euclidean'
kNN	n_neighbors = 60	method = 'mean'
k^{th}NN	n_neighbors = 60	method = 'largest'
OCSVM	nu = 0.008	kernel = 'rbf'
COF	n_neighbors = 60	N/A
ABOD	n_neighbors = 60	N/A
iForest	n_estimators = 100	max_features = 1.0
HBOS	n_bins = 90	tolerance = 0.5
COPOD	N/A	N/A
PCA	n_components = 'mle'	svd_solver = 'full'
VAE	epochs = 500	hidden layers, as described in (8)
SO-GAAL	epochs = 25	N/A
MO-GAAL	epochs = 25	N/A

Single-Objective Generative Adversarial Active Learning (SO-GAAL), and Multi-Objective Generative Adversarial Active Learning (MO-GAAL) [55].

The VAE's architecture was designed individually for each dataset. The number and dimensions of the hidden layers were determined by the number of features in each dataset. Given that K represents the number of features in a given dataset, the hidden layers were built as described in (8).

$$\begin{cases} K \cdot [0.75; 0.5; 0.33; 0.25] & \text{if } K \geq 100 \\ K \cdot [0.75; 0.5; 0.25] & \text{if } 100 > K \geq 50 \\ K \cdot [0.5; 0.25] & \text{if } 50 > K \geq 6 \\ 2 & \text{if } K < 6 \end{cases} \tag{8}$$

The implementation from the Python PyOD package [56] was used to perform the evaluation of the algorithms.

3) Meta-model

A key part of a meta-learner is a meta-model, whose goal is to select the AD algorithm that performs best on an unseen and unlabelled AD dataset. The meta-model proposed in this study (m_1) is based on a neural network architecture. Neural networks have proven successful in a variety of tasks and are relatively time-efficient when training for small-size problems.

The architecture proposed in this work features three hidden layers (64, 64, and 32 nodes), a dropout of 0.2 after each layer, and the predictive regression layer that outputs the predicted performance values $\hat{y}$ for each algorithm $_1; \ldots; _L$, with L being the number of algorithms used in the meta-training.

The training of each meta-model has been started from random weights and run through up to 1000 epochs. The Early Stopping functionality from Keras library [57] has been implemented to cease further training when no improvement is observed, as measured by the loss on the validation data. Subsequently, the optimal weights have been obtained based on the validation loss. The resulting training length ranged for different meta-learner variants (variants described in Section III-C) from 300 to 1000 epochs. Fig. 2 presents an example of the training process picturing the training

each candidate dataset x_i producing the predicted performance metrics $\hat{y}_{ij}$ for each algorithm α_j. The performance metrics of meta-learners' selected algorithms $\alpha^{SEL} \in A$ were then analysed using a mixed model analysis, where x_i was considered to be the subject. The algorithm α^{SEL} was selected from $\alpha_j \in A$ for each x_i to maximise the predicted performance $\hat{y}_{ij}$. The steps performed are described in Procedure 2 and 3, whereas the functional form for this approach is shown in (9) and (10). For each $p, q, r \in \{1, 2\}$:

$$\hat{y}_{ij} = \hat{y}_i(\alpha_j) = m \circ F(x_i) \tag{9}$$

$$y_i^{SEL} = y_i(\alpha^{SEL}) : \hat{y}_i(\alpha^{SEL}) := \max(\hat{y}_{ij}) \tag{10}$$

where

$$\begin{cases} m = m_p \\ F = F_q \\ \alpha_j \in A_r \end{cases}$$

The indices p, q, r have been omitted in (9) and (10) for readability purposes.

FIGURE 2: Training loss ($loss$) and validation loss (val_loss) versus epochs in training one of the NN-based meta-model variant.

TABLE 3: Summary of the factor components of meta-learners compared in this study.

	Current study	UOMS
Meta-model m_p	NN	CF
Meta-features F_q	19 features specific to AD problems	200 features, combined: statistical and landmarking features
Set of AD algorithms A_r	13 algorithms without HO	298 models: 8 algorithms combined with sets of hyper-parameters

and validation loss in subsequent epochs. The batches of 32 samples were used in training and the Adam optimiser was employed to minimise the mean squared error.

The network architecture parameters, such as the number of hidden layers, the number of nodes, dropout level, epoch count, and batch size, were chosen using the grid search approach and optimised with the Weights and Biases tool [58]. The search for the optimal architecture ranged from 2-layered networks of 4 + 4 nodes to 3-layered networks of 64 + 64 + 64 nodes. More complex architectures were not considered to avoid the risk of model overfitting.

C. EVALUATION STRATEGY

A 2^3-factorial design was implemented to examine the factors which contributed to the performance of meta-learners for unsupervised AD. The factors used in this experiment (meta-feature generation strategies F_q, the base set of AD algorithms A_r, and the meta-model m_p) were from the approach proposed in the current study and the UOMS [18]. Table 3 summarises the factors of both approaches. The experimental design is illustrated in Fig. 3.

To perform the experiment, eight meta-learners $M_s :=$ M_{pqr}, were designed, where $s = 1, \ldots, 8$, and $p, q, r \in \{1, 2\}$, which incorporated two variants of mentioned factors F_q, A_r, and m_p. Indices 1 and 2 have been used to denote factors from the current study and UOMS, respectively. Each combination of the 2^3-factorial design was implemented on

Procedure 2 Training of meta-models

Input:
$\quad F_q \in \mathbb{R}^{N \times K_q}$
$\quad Y_{r\,AUC}, Y_{r\,AP} \in \mathbb{R}^{N \times L_r}$, where $Y_r = Y(A_r)$
Output:
$\quad Y_{AUC}^{SEL}, Y_{AP}^{SEL} \in \mathbb{R}^{N^{test}}$
1: for all $\mu \in \{AUC, AP\}$ do
2: $\quad$ for all $F_q : q \in \{1, 2\}$ do
3: $\quad\quad$ for all $Y_r : r \in \{1, 2\}$ do
4: $\quad\quad\quad$ for all $m_p : p \in \{1, 2\}$ do
$\quad\quad\quad\quad$ {Split the input and output data}
5: $\quad\quad\quad\quad (F_q^{train}; F_q^{test}) \leftarrow F_q$
6: $\quad\quad\quad\quad (Y_r^{train}; Y_r^{test}) \leftarrow Y_r$
7: $\quad\quad\quad\quad m_p \leftarrow train(F_q^{train}; Y_r^{train})$ {Train the meta-model in a supervised manner}
8: $\quad\quad\quad\quad \hat{Y}_r^{test} \leftarrow m_p(F_q^{test})$ {Predict}
9: $\quad\quad\quad\quad$ Procedure 3 {Select the best algorithm}
10: $\quad\quad\quad$ end for
11: $\quad\quad$ end for
12: $\quad$ end for
13: end for
14: return $Y_{AUC}^{SEL}; Y_{AP}^{SEL}$

Figure 4 depicts the meta-model framework, including the multi-factor response, as described in (9).

1) Meta-learner training

The meta-model training in this study was conducted using a typical supervised machine-learning pipeline, where the datasets x_i were split in a 60:15:25 ratio into the train, validation, and test sets. The meta-model of each meta-learner M_l was trained to find a mapping between the relevant meta-features and the AD model performance metrics. The

FIGURE 3: Experimental design including the meta-learner framework with reference to Rice's representation (outlined with a purple line) and factor components used in comparative evaluation (blue stickers) A_1, F_1, m_1 and A_2, F_2, m_2 relate and refer to components proposed in the current study and UOMS, respectively.

FIGURE 4: Meta-model framework applied and the multi-factor response.

Procedure 3 Find the best predicted algorithm's performance

Input:
 $\hat{Y}^{test} \in R^{N^{test} \times L}$
Output:
 $Y^{SEL} \in R^{N^{test}}$
 1: for $i = 1$ to N^{test} do
 2:   $y_i^{SEL} \leftarrow y_i(\cdot^{SEL}) : \hat{y}_i(\cdot^{SEL}) := \max(\hat{y}_{ij})$
 3: end for
 4: return Y^{SEL}

---

validation set was used to help inform the decision to end the training process. The learnt mapping was then applied to the new set of test datasets x_i^{test} where the performance values of each AD algorithm $\hat{y}_{ij}$ were predicted.

2) Comparison of two meta-learners

The performance measure of the meta-learner M on the dataset x_i has been defined as the actual performance y_i^{SEL} (either AUC or AP) of the selected algorithm $\cdot^{SEL}$, as expressed in (10). The value of such a metric would be accordingly higher as the selection got closer to the actual best performing algorithm, reflecting the meta-learner's performance. Another metric used in this study to evaluate meta-learner performance was the meta-learner error D_i, which, for the dataset x_i has been defined as a difference between the best measured performance y_i^{TOP} and the performance of the algorithm selected by meta-learner y_i^{SEL} as shown in (11):

$$D_i = y_i^{TOP} - y_i^{SEL}. \qquad (11)$$

A direct comparison of the meta-learning approaches from the current study to UOMS [18] has been performed. The mean performance $\bar{y}^{SEL}$ measured as AUC and AP over the

set of test datasets, and the meta-learners' mean error $\overline{D}$ have been compared. The statistical significance has been measured using the *Paired t-Test* (paired difference test). In addition to statistical significance, the practical significance (effect size) has been assessed using Cohen's d [59] as outlined in (12) and (13):

$$d_y = \frac{\overline{y}_1^{SEL} - \overline{y}_2^{SEL}}{s_y} \qquad (12)$$

$$d_D = \frac{\overline{D}_1 - \overline{D}_2}{s_D}; \qquad (13)$$

where subscripts 1 and 2 indicate approaches from this study and UOMS, respectively, and s_y and s_D are the pooled standard deviation of 1 and 2 distributions of performance values and errors, respectively. The use of the effect size was motivated by the large sample sizes. The number of observations in such cases makes the variables appear statistically significant. As a result, practical significance is a more useful statistic to recognise. The strength of an effect can be categorized as follows [59]:

small effect $0.2 <$ medium effect $0.5 <$ large effect

3) Statistical analysis

In this study, the factors $Z_c \in \{m_p; F_q; A_r\}$, $c = 1; \dots; 3$, contributing to the error of the meta-learners in choosing the correct algorithm were examined using a mixed model analysis [60]–[62]. The error of the meta-learner or the Distance from the Top, D_{il}, has been defined as the difference between the measured performance of the highest performing AD model y_i^{TOP} on the dataset x_i and the selected algorithm performance metric y_{il}^{SEL} for a specific meta-learner M_l as outlined in (14):

$$D_{il} = \ln\left(y_i^{TOP} - y_{il}^{SEL}\right) : \qquad (14)$$

Thirty principal components $V_{1i}; \dots; V_{ni}$ (which explain 93 % of the variance) were generated from both sets of meta-features and used as covariates to adjust for variability due to the differences between datasets.

The model used is expressed in (15)–(17):

$$\begin{aligned} D_{il} = {}&\beta_0 + \beta_1 Z_1 + \beta_2 Z_2 + \beta_3 Z_3 + \\ &+ \beta_{10} V_{1i} + \dots + \beta_{n0} V_{ni} + \gamma_{0i} + \varepsilon_{il} \end{aligned} \qquad (15)$$

with

$$\gamma_{0i} \sim N\left(0; \sigma^2\right) \qquad (16)$$

$$\varepsilon_{il} \sim N\left(0; \sigma_{\varepsilon}^2\right) \qquad (17)$$

where $\beta_0; \dots; \beta_{n0}$ describe the fixed effects [60], and γ_{0i} expresses the random effects' intercepts. Fixed coefficients $\beta_0; \dots; \beta_3$ were assessed for significance using the F-test with statistical significance set at $p < 0.05$. The effect

size (Cohen's d) of the three components Z_c was calculated using (18):

$$d_c = \beta_p \frac{\beta_c}{\sigma_x^2 + \sigma_\varepsilon^2}; \qquad (18)$$

where $c = 1; \dots; 3$ and β_c represents the fixed parameters estimates of Z_c, as in (15), and σ_x and σ_ε represent the variance of the random components and the error term, respectively, as expressed in (16) and (17).

This section compares the performance results of the meta-learner proposed in this study with UOMS. Subsequently, a mixed model analysis of the factors contributing to the algorithm choice within the eight meta-learners M_l is given.

A. PROPOSED APPROACH VERSUS UOMS

The performance comparison of the two solutions is presented in Table 4. It lists the mean performance $\overline{y}^{SEL}$ measured as AUC and AP obtained over the set of test datasets x_i, and the meta-learners' mean error $\overline{D}$ as defined in (11). In both cases, AUC and AP, the solution proposed in the current study has statistically significant higher mean performance than UOMS ($p < 0.001$ for AUC, $p = 0.028$ for AP).

B. STATISTICAL ANALYSIS

A performance summary of eight meta-learners is presented in Table 5. Interestingly, the hybrid approach of F_q, A_2, and m_1 gives the best performance results, for both meta-learner series, AUC and AP-based.

Table 6 presents the Type III main effects for the mixed model analysis. Whereas the choice of each component Z is statistically significant ($p < 0.001$), the choice of the meta-model has the largest effect size. This is particularly visible when using AUC as a performance metric. The AP metric did not demonstrate a notable difference due to its "lower resolution" (highly skewed distributions: skewness − 2.647, kurtosis − 7.064). The skewed distribution of AP values is expected as unbalanced datasets generally show this behaviour [63], [64]. The NN meta-learner did, however, demonstrate a favourable performance in comparison to the CF group (Table 5).

The interaction terms between the main effects outlined in (15) had no statistical significance and were subsequently excluded from the final statistical model.

C. TIME ANALYSIS

Time analysis was performed on a subset of 20 datasets chosen at random from the entire set used in the current study. The datasets ranged in size from 68 to 5,186 observations and 5 to 147 features. This analysis has been restricted to the end-user perspective, which includes the generation of dataset meta-features and the prediction of the best-suited algorithm. The time summary of both approaches to generate meta-features and perform prediction is presented in Table 7 and Table 8, respectively.

TABLE 4: Comparison of the mean performance $\bar{y}^{SEL}$ and mean error $\bar{D}$ with standard deviations across test datasets of two analysed meta-learner approaches, for AUC and AP.

		Current study		UOMS		D.f.	T-stat.	p-value	Effect size
AUC	$\bar{y}^{SEL}$	0.6703	0.1820	0.6464	0.1940	2317	7.882	<0.001	0.126
	$\bar{D}$	0.1567	0.1315	0.1804	0.1558	2317	-7.882	<0.001	0.162
AP	$\bar{y}^{SEL}$	0.1413	0.1955	0.1369	0.1905	2302	2.193	0.028	0.009
	$\bar{D}$	0.1685	0.1466	0.1749	0.1543	2302	-2.193	0.028	0.042

D.f. – degrees of freedom, T-stat. – t-test statistic

TABLE 5: Mean performance $\bar{y}^{SEL}$ with standard deviations across test datasets of eight meta-learners for AUC and AP.

Meta-model	AD alg.	Meta-features	$\bar{y}^{SEL}$, AUC		$\bar{y}^{SEL}$, AP	
M_1 (NN)	A_2	F_2	0.692	0.177	0.154	0.215
M_1 (NN)	A_2	F_1	0.679	0.179	0.150	0.209
M_1 (NN)	A_1	F_2	0.678	0.183	0.146	0.200
M_1 (NN)	A_1	F_1	0.670	0.182	0.141	0.195
M_2 (CF)	A_2	F_2	0.646	0.194	0.137	0.191
M_2 (CF)	A_2	F_1	0.630	0.202	0.136	0.192
M_2 (CF)	A_1	F_2	0.630	0.191	0.127	0.176
M_2 (CF)	A_1	F_1	0.616	0.193	0.120	0.169

TABLE 6: Type III analysis of the main effects of the meta-learner components.

	Component	D.f.	F-stat.	p-value	Effect size
AUC	Intercept	1,2291	13382.78	<0.001	
	Meta-features F_q	1,16244	51.457	<0.001	0.082
	AD models A_r	1,16244	127.648	<0.001	0.130
	Meta-model m_p	1,16244	678.938	<0.001	0.300
AP	Intercept	1,2294	14091.410	<0.001	
	Meta-features F_q	1,16009	0.139	0.710	0.004
	AD models A_r	1,16025	69.087	<0.001	0.095
	Meta-model m_p	1,16016	111.051	<0.001	0.121

D.f. – degrees of freedom, F-stat. – F-test statistic

TABLE 7: Time in seconds to generate dataset meta-features summarised for a random sample set of 20 datasets.

Statistic (time, s)	UOMS	Current study
Mean	0.886	2.265
St. dev.	0.777	3.627
Min	0.351	0.044
Max	3.103	10.517
Cases with shorter time	8	12

TABLE 8: Time in seconds to predict the best performing algorithm summarised for a random sample set of 20 datasets and across the meta-learner variants.

Statistic (time, s)	CF	NN
Mean	1.376	0.492
St. dev.	0.574	0.114
Min	0.914	0.400
Max	4.537	0.956

Although the UOMS approach takes less time on average to generate the meta-feature set, the number of datasets for which the generation takes less time is greater for the currently presented approach. The current approach is more time-consuming for datasets with a relatively higher number of observations because it involves calculating the distances between all instances within a dataset. Ultimately, the time difference between UOMS and the current approach was not statistically significant for the measured sample set (t_{19} = 1:835, p = 0:082).

When compared to the CF meta-model, the prediction times are shorter on average and more consistent with the use of NN. Furthermore, training times for CF models were significantly longer than for NN-based models (approx. 20 hours versus approx. 10 minutes, per meta-learner variant).

The above analysis was carried out on the machine with the following subcomponents: 1.6 GHz Dual-Core Intel Core i5 processor and 8 GB of 2133 MHz RAM.

This research performed a direct comparison between a new method proposed here with the UOMS approach [18]. The results demonstrate that while there was a statistically significant improvement with the proposed method, there was a negligible effect (practical difference) when comparing the AP mean error (Cohen's d = 0.042) and a small effect when comparing the AUC mean error (Cohen's d = 0.162) between the two approaches (Table 4). The proposed method in this study demonstrates, however, that equivalent results can be obtained from a substantially reduced feature set and the omission of HO in the meta-learning configurations. Previous works have assumed that HO was the main characteristic in meta-learning. Whereas the initial results are only a direct comparison between the method proposed in the current study and the UOMS, the mixed model analysis helps to elucidate the characteristics that have a contributing effect.

The mixed model analysis, summarised in Table 6, shows that for AUC, the larger meta-feature set F_2 provides only a marginal benefit over the smaller set of meta-features F_1 (Cohen's d = 0.082). It is worth noting that while F_2 makes extensive use of generic statistical features, the compact set F_1 is crafted to reflect anomaly characteristics. When comparing differences in the performance between the two sets of AD models, the large set with HO (A_2) outperforms

the small set without HO (A_1). However, given the number of models in both groups (298 versus 13), the effect size is not as compelling as one would expect (Cohen's d 0.130). The analysis demonstrates that the choice of meta-model has the most significant impact on the meta-learner's final performance, with the NN-based meta-model outperforming the state of the art CF approach (B_2 Cohen's $d = 0.300$).

This outcome is an important consideration given that current AutoML or meta-learning studies frequently direct their attention to other aspects, such as meta-features development [19], [31] or HO [23], [24], [26]. This work demonstrates that investing time and effort into creating an adequate meta-model that can successfully utilise data from historical evaluations is the most promising approach for improving meta-learners for unsupervised AD.

The contributions measured on AP show a similar pattern, however, the effects are less visible. With a significance level at 0.05, the influence of the meta-features is not statistically significant. Consequently, the effect size is negligible. The contributions of the other two components are larger, but their effect sizes on the AP metric are also minimal.

The results of the time analysis show that using a meta-learner within an AD pipeline outweighs the costs in terms of time and computing resources. For a dataset with 1,000 observations and 45 features, the extra time of 1-2 seconds for meta-feature generation and 0.5 seconds for finding the best suited algorithm could potentially save hours on a trial-and-error process of finding the best performing algorithm and evaluating the results.

In this study, a new meta-learner to help in the identification of an appropriate unsupervised machine learning algorithm when applied to anomaly detection problems was presented. When compared to the current state of the art, UOMS [18], the proposed method demonstrated a statistically significant improvement in results using two performance metrics, AUC and AP.

In addition, using a 2^3 experimental design, an experiment was conducted to understand which of the component parts (meta-model, meta-features, algorithm set) in both approaches had the greatest influence on the overall performance of the meta-learner. While the choice of meta-features and the base set of AD algorithms were shown to have a relatively small effect size, the meta-model choice had the largest effect on the meta-learner performance. Furthermore, the analysis revealed that a hybrid version of both approaches, UOMS and the one proposed here, gave the best performance results.

Finally, the experiments in this study were carried out on the largest number of datasets used to date for examining an algorithm selection for an unsupervised anomaly detection task.

Future work in this area should focus on an in-depth analysis of approaches used in the meta-models and the development of datasets that represents a truly robust test to any future meta-learners.

[1] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," *Data mining and knowledge discovery*, vol. 30, pp. 891–927, 2016.

[2] H. Wang, M. J. Bah, and M. Hammad, "Progress in outlier detection techniques: A survey," *Ieee Access*, vol. 7, pp. 107 964–108 000, 2019.

[3] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *CoRR*, vol. abs/1901.03407, 2019. [Online]. Available: http://arxiv.org/abs/1901.03407

[4] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K. R. Muller, "A Unifying Review of Deep and Shallow Anomaly Detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, May 2021.

[5] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLOS ONE*, vol. 11, no. 4, p. e0152173, Apr 2016. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0152173

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, Jul 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[7] E. Schubert, A. Zimek, and H.-P. Kriegel, "Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection," *Data mining and knowledge discovery*, vol. 28, no. 1, pp. 190–237, 2014.

[8] C. Wang, H. Gao, Z. Liu, and Y. Fu, "A new outlier detection model using random walk on local information graph," *IEEE Access*, vol. 6, pp. 75 531–75 544, 2018.

[9] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[10] R. Ali, S. Lee, and T. C. Chung, "Accurate multi-criteria decision making methodology for recommending machine learning algorithm," *Expert Systems with Applications*, vol. 71, pp. 257–278, 2017.

[11] B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Fréchette, H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney *et al.*, "Aslib: A benchmark library for algorithm selection," *Artificial Intelligence*, vol. 237, pp. 41–58, 2016.

[12] I. Khan, X. Zhang, M. Rehman, and R. Ali, "A literature survey and empirical study of meta-learning for classifier selection," *IEEE Access*, vol. 8, pp. 10 262–10 281, 2020.

[13] C. Lemke, M. Budka, and B. Gabrys, "Metalearning: a survey of trends and technologies," *Artificial intelligence review*, vol. 44, no. 1, pp. 117–130, 2015.

[14] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[15] K. A. Smith-Miles, "Cross-disciplinary perspectives on meta-learning for algorithm selection," *ACM Computing Surveys (CSUR)*, vol. 41, no. 1, pp. 1–25, 2009.

[16] J. R. Rice, "The algorithm selection problem," in *Advances in computers*. Elsevier, 1976, vol. 15, pp. 65–118.

[17] S. Kandanaarachchi, M. A. Munoz, and K. Smith-Miles, "Instance space analysis for unsupervised outlier detection," *EDML@ SDM*, 2019, pp. 32–41.

[18] Y. Zhao, R. Rossi, and L. Akoglu, "Automatic unsupervised outlier model selection," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4489–4502, 2021.

[19] M. Kotlar, M. Punt, Z. Radivojević, M. Cvetanović, and V. Milutinović, "Novel meta-features for automated machine learning model selection in anomaly detection," *IEEE Access*, vol. 9, pp. 89 675–89 687, 2021.

[20] V. Papastefanopoulos, P. Linardatos, and S. Kotsiantis, "Unsupervised outlier detection: A meta-learning algorithm based on feature selection," *Electronics*, vol. 10, no. 18, p. 2236, 2021.

[21] C. Le Clei, Y. Pushak, F. Zogaj, M. O. Kareshk, Z. Zohrevand, R. Harlow, H. F. Moghadam, S. Hong, and H. Cha, "N-1 experts: Unsupervised anomaly detection model selection," in *First Conference on Automated Machine Learning (Late-Breaking Workshop)*, 2022.

[22] J. Vanschoren, "Meta-learning," in *Automated machine learning*. Springer, Cham, 2019, pp. 35–61.

[23] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn: automatic hyperparameter con guration for scikit-learn," in *ICML workshop on AutoML*, vol. 9. Citeseer, 2014, p. 50.

[24] M. Feurer, J. Springenberg, and F. Hutter, "Initializing bayesian hyperparameter optimization via meta-learning," *Proceedings of the AAAI Conference on Arti cial Intelligence*, vol. 29, no. 1, 2015.

[25] R. G. Mantovani, A. L. Rossi, J. Vanschoren, B. Bischl, and A. C. Carvalho, "To tune or not to tune: recommending when to adjust svm hyperparameters via meta-learning," *2015 International Joint Conference on Neural Networks (IJCNN)* Ieee, 2015, pp. 1–8.

[26] T. Horváth, R. G. Mantovani, and A. C. de Carvalho, "Effects of random sampling on svm hyper-parameter tuning," *International Conference on Intelligent Systems Design and Applications* Springer, 2016, pp. 268–278.

[27] S. Sanders and C. G. Giraud-Carrier, "Informing the use of hyperparameter optimization through metalearning," *2017 IEEE International Conference on Data Mining (ICDM)* pp. 1051–1056, 2017.

[28] M. Wistuba, N. Schilling, and L. Schmidt-Thieme, "Two-stage transfer surrogate model for automatic hyperparameter optimization," *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16* Springer, 2016, pp. 199–214.

[29] R. G. Mantovani, A. L. Rossi, E. Alcobaça, J. Vanschoren, and A. C. de Carvalho, "A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves svm classi ers," *Information Sciences* vol. 501, pp. 193–221, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002002551930533X

[30] R. G. Mantovani, A. L. D. Rossi, E. Alcobaça, J. C. Gertrudes, S. B. Junior, and A. C. P. de Leon Ferreira de Carvalho, "Rethinking default values: a low cost and ef cient strategy to de ne hyperparameters," *ArXiv*, vol. abs/2008.00025, 2020.

[31] J. Kanda, A. De Carvalho, E. Hruschka, C. Soares, and P. Brazdil, "Meta-learning to select the best meta-heuristic for the traveling salesman problem: A comparison of meta-features," *Neurocomputing* vol. 205, pp. 393–406, 2016.

[32] Y. Abdrashitova, A. Zabashta, and A. Filchenkov, "Spanning of meta-feature space for travelling salesman problem," *Procedia Computer Science* vol. 136, pp. 174–182, 2018, 7th International Young Scientists Conference on Computational Science, YSC2018, 02-06 July2018, Heraklion, Greece. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918315552

[33] A. E. Gutierrez-Rodríguez, S. E. Conant-Pablos, J. C. Ortiz-Bayliss, and H. Terashima-Marín, "Selecting meta-heuristics for solving vehicle routing problems with time windows via meta-learning," *Expert Systems with Applications* vol. 118, pp. 470–481, 2019.

[34] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Arti cial intelligence review* vol. 18, no. 2, pp. 77–95, 2002.

[35] M. A. Muñoz, Y. Sun, M. Kirley, and S. K. Halgamuge, "Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges," *Information Sciences* vol. 317, pp. 224–245, 2015.

[36] I. Guyon, L. Sun-Hosoya, M. Boullé, H. J. Escalante, S. Escalera, Z. Liu, D. Jajetic, B. Ray, M. Saeed, M. Sebag *et al.*, "Analysis of the automl challenge series," *Automated Machine Learning* p. 177, 2019.

[37] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-weka: Automatic model selection and hyperparameter optimization in weka," in *Automated machine learning* Springer, Cham, 2019, pp. 81–95.

[38] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Ef cient and robust automated machine learning," *Advances in neural information processing systems* vol. 28, 2015.

[39] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-sklearn 2.0: Hands-free automl via meta-learning," *The Journal of Machine Learning Research* vol. 23, no. 1, pp. 11 936–11 996, 2022.

[40] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, M. Urban, M. Burkart, M. Dippel, M. Lindauer, and F. Hutter, "Towards automatically-tuned deep neural networks," in *Automated machine learning*. Springer, Cham, 2019, pp. 135–149.

[41] R. S. Olson and J. H. Moore, "Tpot: A tree-based pipeline optimization tool for automating machine learning," in *Workshop on automatic machine learning*. PMLR, 2016, pp. 66–74.

[42] C. Steinruecken, E. Smith, D. Janz, J. Lloyd, and Z. Ghahramani, "The automatic statistician," in *Automated Machine Learning* Springer, Cham, 2019, pp. 161–173.

[43] S. Kandanaarachchi, M. A. Muñoz, R. J. Hyndman, and K. Smith-Miles, "On normalization and algorithm selection for unsupervised outlier detection," *Data Mining and Knowledge Discovery* vol. 34, no. 2, pp. 309–354, Mar 2020. [Online]. Available: https://doi.org/10.1007/s10618-019-00661-z

[44] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data* 2000, pp. 93–104.

[45] S. Ramaswamy, R. Rastogi, and K. Shim, "Ef cient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data* 2000, pp. 427–438.

[46] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," *Advances in neural information processing systems* 2000, pp. 582–588.

[47] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," *Paci c-Asia conference on knowledge discovery and data mining* Springer, 2002, pp. 535–548.

[48] H.-P. Kriegel, M. Schubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* 2008, pp. 444–452.

[49] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining* IEEE, 2008, pp. 413–422.

[50] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm," *KI-2012: poster and demo track* vol. 9, 2012.

[51] T. Pevny, "Loda: Lightweight on-line detector of anomalies," *Machine Learning* vol. 102, no. 2, pp. 275–304, 2016.

[52] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu, "Copod: copula-based outlier detection," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 1118–1123.

[53] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classi er," in Proceedings of the IEEE foundation and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM03) Jan 2003, pp. 172–179.

[54] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114* 2013.

[55] Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang, and X. He, "Generative adversarial active learning for unsupervised outlier detection," *IEEE Transactions on Knowledge and Data Engineering* vol. 32, no. 8, pp. 1517–1528, 2019.

[56] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research* vol. 20, no. 96, pp. 1–7, 2019. [Online]. Available: http://jmlr.org/papers/v20/19-011.html

[57] F. Cholletet al., "Keras," https://keras.io, 2015.

[58] L. Biewald, "Experiment tracking with weights and biases," 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/

[59] J. Cohen, *Statistical power analysis for the behavioral sciences* Academic press, 2013.

[60] W. Stroup, Generalized Linear Mixed Models: Modern Concepts, Methods and Applications ser. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2012. [Online]. Available: https://books.google.ie/books?id=GcGrySpkXRMC

[61] C. McCulloch and S. Searle, *Generalized, Linear, and Mixed Models* ser. Wiley Series in Probability and Statistics. Applied Probabil. Wiley, 2004. [Online]. Available: https://books.google.ie/books?id=bWDPukohugQC

[62] E. Demidenko, *Mixed models: theory and applications with R* John Wiley & Sons, 2013.

[63] H. Haibo and M. Yunqian, "Imbalanced learning: foundations, algorithms, and applications," *Wiley-IEEE Press* vol. 1, p. 27, 2013.

[64] R. Viola, L. Gautheron, A. Habrard, and M. Sebban, "Metaap: A meta-tree-based ranking algorithm optimizing the average precision from imbalanced data," *Pattern Recognition Letters* vol. 161, pp. 161–167, 2022.