Applied Intelligence (2025) 55:875
https://doi.org/10.1007/s10489-025-06706-9

Automated machine learning for positive-unlabelled learning

Jack D. Saunders[1] · Alex A. Freitas[1]

Accepted: 8 June 2025

Abstract
Positive-Unlabelled (PU) learning is a field of machine learning that aims to learn classifiers from data consisting of labelled positive and unlabelled instances, which can be in reality positive or negative, but whose label is unknown. Many PU learning methods have been proposed over the last two decades, so many so that selecting an optimal method for a given PU learning task presents a challenge. Our previous work has addressed this by proposing GA-Auto-PU, the first Automated Machine Learning (Auto-ML) system for PU learning. In this work, we propose two new PU learning Auto-ML systems: BO-Auto-PU, based on a Bayesian Optimisation (BO) approach, and EBO-Auto-PU, based on a novel evolutionary/BO approach. We present an extensive evaluation of the three Auto-ML systems, comparing them to each other and to well-established PU learning methods across 60 datasets (20 datasets, each with 3 versions). The results of the comparison show statistically significant improvements in predictive accuracy over the baseline methods, as well as large improvements in computational time for the newly proposed Auto-PU systems over the original Auto-PU system.

Keywords Positive-unlabelled learning · Automated Machine Learning (Auto-ML) · Bayesian optimisation · Genetic algorithm · Classification

1 Introduction

Positive-Unlabelled (PU) learning is a growing field of machine learning that aims to learn classifiers from data consisting of a set of labelled positive instances and a set of unlabelled instances which can be either positive or negative, but whose label is unknown [1]. This is an important field because, in real-world applications, obtaining a fully labelled dataset is often very expensive or impractical [1].

To address this issue, many methods have been proposed to learn classifiers from partially labelled data [1, 2]; including semi-supervised classification methods that learn from data with relatively small subsets of positive and negative instances, where most instances are unlabelled.

PU learning is a specialised case of semi-supervised learning where, among the truly positive-class instances, only a subset is labelled as positive; whilst all the other instances are unlabelled– i.e., it is unclear if they are positive or negative instances [1]. Thus, positive-unlabelled learning presents an arguably greater challenge than the standard semi-supervised learning paradigm due to the complete absence of negative labels in the data.

There are many real-world applications of PU learning, including cybersecurity [3, 4], bioinformatics [5–7], and text mining [8–10]. For example, PU learning has been used to predict disease-related genes [6]. In this task, disease-associated genes are the labelled positive instances, as confirmed by biomedical experiments. However, the vast majority of genes thought not to be associated with diseases have not undergone such experiments, as such experiments are expensive. Hence, genes not associated with diseases are better thought of as unlabelled instances, as there is no experimental evidence for assigning them any class label. An example involving text mining is found in [10], which proposed a PU learning method for web page classification. Scraping web pages is an easy and quick task, so assembling a large dataset is a simple process. However, the majority of the instances (webpages) will be unlabelled as manually labelling each instance is an expensive task. As illustrated by these examples, PU learning is appropriate when the

✉ Jack D. Saunders
jsaunders0027@gmail.com

Alex A. Freitas
A.A.Freitas@kent.ac.uk

[1] School of Computing, University of Kent, Canterbury CT2 7FS, Kent, United Kingdom

dataset consists of a small sample of reliable positives and a larger remaining sample of unknown-label instances.

The most popular approach to PU learning is the two-step approach [1], which aims, in step 1, to find a set of reliable negative instances in the unlabelled set. That is, a set of instances that are substantially different from the positive set and are likely not unlabelled positive instances. In step 2, a classifier is trained to distinguish between labelled positive instances and reliable negative instances, building a classifier which, if the reliable negative set is accurate, can classify an unseen instance as either positive or negative.

Many PU-learning methods have been proposed, and so an exhaustive search of all to find the optimal method for a given task is unfeasible. To address this issue, GA-Auto-PU, based on a Genetic Algorithm (GA) was proposed as the first Automated Machine Learning (Auto-ML) system for PU learning [11]. GA-Auto-PU outperformed baseline PU learning methods, including a state-of-the-art approach based on deep forests [12]. However, the system was very computationally expensive, making it potentially impractical for many users. For this reason, this work introduces BO-Auto-PU, a Bayesian Optimisation-based Auto-ML system for PU learning, and EBO-Auto-PU, an Evolutionary Bayesian Optimisation-based Auto-ML system for PU learning (a hybrid of GA-Auto-PU and BO-Auto-PU). As shown in Section 6, both are much more computationally efficient than GA-Auto-PU and achieve good predictive performance.

This paper is organised as follows. Section 2 details the background on PU learning and Auto-ML. Section 3 outlines the Auto-ML framework for PU learning. Section 4 details the three Auto-PU systems, with pseudocodes of the new BO-Auto-PU and EBO-Auto-PU. Section 5 outlines our experimental methodology and the datasets used in the experiments. Section 6 details the results, comparing the three Auto-ML systems and two baseline PU learning methods. Section 7 concludes this work.

2 Background

2.1 Positive-Unlabelled (PU) learning

Positive-Unlabelled (PU) learning is a specialised case of semi-supervised classification that involves training a classifier to distinguish between the positive and negative class, given only positive and unlabelled instances [1]. The aim is to predict an instance's probability of belonging to the positive class, P(y=1), given information about whether that instance is labelled, P(s=1). Since each instance is annotated as "labelled positive" or "unlabelled", 1 or 0, a standard machine learning model will predict those two annotations (positive or unlabelled), rather than whether an instance is positive or negative. Therefore, to accurately predict an instance's class as positive or negative, alternative approaches must be followed in the place of standard binary classification.

The most popular PU learning approach is the two-step framework, consisting of a first step to identify reliable negative instances and a second step to learn a classifier that can distinguish the labelled positives and the reliable negatives [1].

In step 1, we generally train a classifier to distinguish between the labelled positive instances and the unlabelled set. Often, this process is completed iteratively, splitting the unlabelled set into multiple sets to handle the class imbalance that is characteristic of many PU learning tasks, where the unlabelled set is usually larger than the positive set [1]. From this process, we learn a classifier that distinguishes between the probability of an instance being labelled, P(s=1), and the probability of an instance being unlabelled, P(s=0). We then assume that those instances with the lowest probability of belonging to the positive class are likely not unlabelled positive instances and are thus treated as the reliable negative set. This assumes smoothness and separability of the data [1]. That is, we assume that similar data instances have similar probabilities of belonging to the positive class (smoothness); and we assume that there is a natural division between positive and negative classes (separability).

Many approaches propose a second stage of the first step, akin to a classic semi-supervised learning method, that involves expanding the reliable negative set by training a classifier to distinguish the labelled positives and the initial reliable negative sets, classifying the remaining unlabelled instances, and adding those unlabelled instances with a low P(y=1) to the reliable negative set.

We refer to the curation of the initial reliable negative set as Phase 1A, and the semi-supervised step as an optional Phase 1B. Note the use of "Phase" rather than "Step". This allows us to discuss the individual steps in each phase without confusion.

The second phase involves learning a classifier to distinguish the labelled positives and the reliable negative instances (identified in Phase 1). Thus, Phase 2 involves learning a classifier that can predict an instance's P(y=1) rather than P(s=1).

Many PU-learning methods have been proposed [1, 2]. One such method is "Spy with Expectation Maximisation" (S-EM), proposed in [8]. S-EM modifies the standard two-step procedure with the introduction of "spy" instances, used to calculate the threshold under which an instance's P(s=1) must fall in order for the instance to be deemed a reliable negative. S-EM was one of the first proposed two-step methods, but it is still considered an effective method and is frequently used as a baseline method when proposing new PU learning methods [13–19].

A recently proposed state-of-the-art two-step PU learning method is the Deep Forest-PU (DF-PU) method [12]. DF-PU uses the powerful deep forest classifier [20] in the standard two-step framework to build a PU learning classifier.

Both S-EM and DF-PU will be used in this work's experiments as baseline methods to evaluate the new Auto-PU systems. The DF-PU method can be created by our Auto-PU systems as it is simply the two-step framework utilising a deep forest classifier. S-EM cannot be directly replicated as it employs the expectation maximisation step, but the Auto-PU systems utilising the extended search space are able to employ the same "spy" methodology.

The assumptions most commonly made by PU learning algorithms are separability, smoothness, and selected completely at random (SCAR) [1]. The separability assumption states that there exists a classifier that can perfectly separate the positive and negative instances in the feature space. The smoothness assumption states that instances which are close to each other in the feature space are likely to belong to the same class. These assumptions are foundational to the two-step approach.

The SCAR assumption, formalised by [21], states that the positive instances are labelled irrespective of their features, and thus the labelled set is an independent and identically distributed sample from the positive distribution. That is, for the given data, Pr(s=1) = Pr(s=1   x), where Pr(s=1) represents the probability of an instance being labelled and x is an instance's feature vector. I.e., the sample of positive instances in the labelled positive set is representative of all truly positive instances, both labelled and unlabelled. Making the SCAR assumption allows us to assume that the instances in the labelled positive set are representative of the instances within the unlabelled set, and thus, if a classifier can accurately predict the labelled positive instances, it should, in theory, be able to predict the unlabelled positive instances also. For this reason, the SCAR assumption is foundational to some PU learning approaches.

The absence of negative instances makes it hard to evaluate PU learning models as predictive accuracy metrics usually rely on knowledge of the true class labels of each instance. However, in PU learning the true class label is known only for a sample of positive instances. The other positive instances, and all negative instances, are unlabelled. Hence, standard accuracy metrics [22] cannot be correctly calculated.

Under the SCAR assumption, since the sample of positive instances in the labelled positive set is representative of all truly positive instances, both labelled and unlabelled, we can estimate performance metrics for models tested on genuine PU data; i.e., PU data that has not been engineered from a standard dataset with positive-negative (PN) labels. However, such performance estimates are not entirely robust. Arguably, a more robust approach is to evaluate a PU learning method on an engineered PU dataset before applying that method to a genuine PU learning task.

As noted in [23], the approach most frequently used in the literature is to evaluate proposed methods on engineered PU data created from a standard PN dataset by hiding a certain percentage of positive instances in the negative set, thus creating an unlabelled set (i.e., all negatives and the hidden positives will be treated as 'unlabelled'). This is done for the training set, whilst leaving the test set untouched. That is, the test set will contain positive and negative instances as in the original dataset. Hence, the model is trained on PU data but evaluated on fully labelled data. Therefore, we can accurately calculate standard PN metrics. This is arguably a more robust approach as the SCAR assumption is not required and we can accurately calculate performance metrics based on the known class labels of the instances in the test set.

2.2 Automated Machine Learning (Auto-ML)

Automated Machine Learning (Auto-ML) is a rapidly growing field of machine learning that aims to limit human involvement in the machine learning pipeline development process by tailoring such a pipeline to a specific input dataset through the use of an optimisation method [24, 25]. This allows users without extensive knowledge of machine learning to operate complex ML pipelines and enables users to find effective pipelines for specific learning tasks without guesswork or arbitrary choices. For a review of Auto-ML approaches, see [24–26].

In this work, we use three types of optimisation approaches for Auto-ML: a genetic algorithm (GA), Bayesian optimisation (BO), and a surrogate-assisted evolutionary algorithm, which we refer to as evolutionary Bayesian optimisation (EBO).

GAs are optimisation methods that rely on the principle of natural selection to evolve candidate solutions based on a fitness (objective) function [27]. In Auto-ML, a candidate solution is usually a machine learning (ML) pipeline, and the objective function is usually a predictive accuracy measure [28, 29]. In the context of Auto-ML for PU learning (hereafter named "Auto-PU"), GA-Auto-PU uses a GA for optimisation of PU learning methods [11, 30].

BO is an iterative model-based optimisation approach that leverages surrogate models to identify areas of the search space that show promise, either due to their potential predictive performance or due to their placement in an unexplored region of the search space, depending on the acquisition function [31]. The acquisition function is a tool to quantity the potential value of evaluating different candidate ML pipelines. The function can be as simple as the value assigned by the surrogate model to the candidate ML pipeline, or a more complex function such as Expected Improvement [32] or Probability of Improvement [33]. Thus, the next candidate solution to evaluate using the objective function is determined by which candidate solution maximises the acquisition function. Generally, the objective function in an Auto-ML setting is expensive as it involves training a ML pipeline. Hence, by selecting a limited number of candidates (often a single candidate solution) to assess with the objective function, computational expense is reduced, making BO a generally much more efficient (faster) optimisation method than a GA.

One potential drawback of BO is relatively low exploration of the search space and the lack of population diversity (in each iteration, usually a single candidate solution is evaluated using the objective function). Both issues are alleviated in GAs through the use of a diverse population and evolutionary operations (crossover and mutation). As such, surrogate-assisted GAs often achieve a good trade-off between the exploration and candidate diversity of a GA and the computational efficiency of BO [34].

In this work, a surrogate model directs the GA towards good search space areas whilst maintaining a population to evolve [34]. Our approach to surrogate-assisted GAs, named Evolutionary Bayesian Optimisation (EBO), is discussed in Section 4.

3 Auto-PU: the proposed Auto-ML framework for positive-unlabelled learning

This section outlines the proposed Auto-ML framework for PU learning, specifying the search spaces (Section 3.1) and the objective function used to evaluate candidate solutions (Section 3.2). Note that the search spaces and the objective function are the same for all three types of Auto-PU systems used in this work, which vary regarding their optimisation method: GA, BO or a hybrid EBO. Hence, this common framework of search spaces and objective function is discussed separately in this section, whilst the next three sections discuss the optimisation method-specific details of each type of Auto-PU system.

3.1 Search spaces

In Auto-ML, a search space is the set of all candidate solutions that can be found by the search algorithm, consisting of a pre-defined set of algorithms with their hyperparameters and their respective values. For our Auto-PU systems (Auto-ML systems for PU learning), the search space is defined by the two-step PU learning framework [1]. That is, a candidate solution is a two-step PU learning method, consisting of Phases 1A, 1B, and 2, discussed in Section 2.1. Each phase has a set of hyperparameters and their candidate values, which define the search space of our Auto-PU systems.

Our experiments have used two variations of the search space, namely the base search space and the extended search space, detailed in Sections 3.1.1 and 3.1.2, respectively.

3.1.1 The base search space

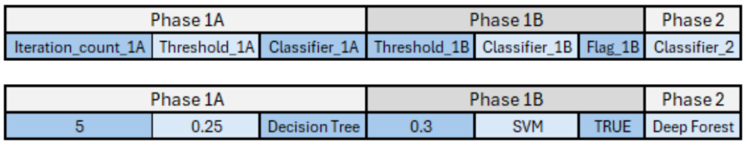
The base search space, proposed in our previous work [11], allows the system to build relatively simple two-step PU learning methods that do not utilise any heuristics for determining the values of the hyperparameters. Specifically, the search space is defined by the following 7 hyperparameters and their corresponding candidate values:

- $Iteration\_count\_1A$: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }
- $Threshold\_1A$: { 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5 }
- $Classifier\_1A$: { Candidate_classifiers }
- $Threshold\_1B$: { 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5 }
- $Classifier\_1B$: { Candidate_classifiers }
- $Flag\_1B$: True, False
- $Classifier\_2$: { Candidate_classifiers }

Where $Candidate\_classifiers$ represents 18 different candidate classification algorithms: Gaussian naïve Bayes, Bernoulli naïve Bayes, Random forest, Decision tree, Multilayer perceptron, Support vector machine, Stochastic gradient descent classifier, Logistic regression, K-nearest neighbour, Deep forest, AdaBoost, Gradient boosting classifier, Linear discriminant analysis, Extra tree classifier, Extra trees (ensemble) classifier, Bagging classifier, Gaussian process classifier, and Histogram-based gradient boosting classification tree.

Together, these hyperparameters form Phase 1A, Phase 1B, and Phase 2 of the two-step PU learning framework described in Section 2.1. Figure 1 shows how these hyperparameters form these phases, showing the structure of a candidate solution in the base search space and an example candidate solution (with examples of hyperparameter values).

Phase 1A consists of the hyperparameters $Iteration\_count\_1A$, $Threshold\_1A$, and $Classifiers\_1A$. $Iteration\_count\_1A$ specifies the number of subsets into which the unlabelled set is split



Phase 1A			Phase 1B			Phase 2
Iteration_count_1A	Threshold_1A	Classifier_1A	Threshold_1B	Classifier_1B	Flag_1B	Classifier_2

Phase 1A			Phase 1B			Phase 2
5	0.25	Decision Tree	0.3	SVM	TRUE	Deep Forest

Fig. 1 Example of a candidate solution in the base search space

for learning a classifier to distinguish between the positive and unlabelled sets, and also the number of iterations that a classification algorithm is run in Phase 1A. E.g., if $Iteration\_count\_1A = 5$, the unlabelled set is split into 5 subsets, each with 20% of the unlabelled data, and the classification algorithm is run 5 times, each with a different subset of unlabelled training instances. This helps to handle the class imbalance often found in PU learning datasets. $Threshold\_1A$ specifies the maximum value of the predicted probability of the positive class for an instance to be considered a reliable negative instance. The $Classifier\_1A$ is simply the classifier used to predict the reliable negative instances.

Phase 1B consists of the hyperparameters $Threshold\_1B$, $Classifier\_1B$, and $Flag\_1B$. $Threshold\_1B$ and $Classifier\_1B$ are analogous to those used in phase 1A. $Flag\_1B$ indicates whether or not to skip phase 1B, which is an optional phase in PU learning methods. Given the similarities between Phase 1A and Phase 1B, a natural question arises as to why we exclude an iteration count parameter from Phase 1B. The reason is that the iteration count parameter was introduced in order to handle the class imbalance inherent to PU learning datasets, but this is not generally an issue once an initial reliable negative set has been created as this set is simply a small subset of the unlabelled set. Furthermore, class imbalance is indirectly handled by the $Threshold\_1A$ parameter, which will probably be adapted to take a smaller value (and thus fewer instances will be added to the reliable negative set) if the reliable negative set becomes large enough to detriment predictive performance.

Phase 2 simply consists of the hyperparameter $Classifier\_2$. This classifier will be trained to distinguish the positive set and the reliable negative set extracted from the unlabelled set in phase 1A and potentially 1B.

The base search space has 11,664,000 candidate solutions [11].

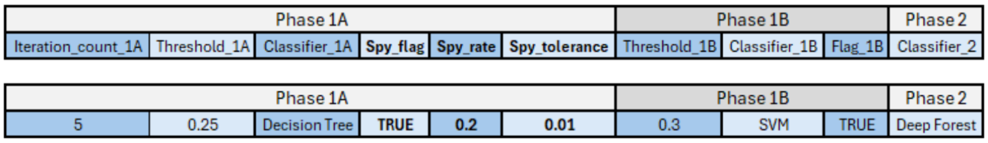
So, in Phase 1A of the example candidate solution shown in Fig. 1, the unlabelled set would be split into 5 subsets (defined by the $Iteration\_count\_1A$ hyperparameter). Each of these subsets in turn, along with the labelled positive set, would be used to train a decision tree classifier ($Classifier\_1A$) which would then predict the probability of the unlabelled instances in the current subset belonging to the positive class. Those instances with a predicted probability of less than 0.25 (the $Threshold\_1A$ hyperparameter) would be added to the reliable negative set and removed from the unlabelled set. Then, as the $Flag\_1B$ parameter is set to True, a support vector machine (SVM) classifier ($Classifier\_1B$) would be built using the labelled positive instances as the positive set and the reliable negative instances identified in Phase 1A as the negative set. It would then be used to classify the remaining unlabelled instances, and those with a predicted probability of belonging to the positive class of less than 0.3 (the $Threshold\_1B$ hyperparameter) would be added to the reliable negative set. Finally, a deep forest classifier ($Classifier\_2$) would be trained on the labelled positive and the reliable negative sets.

3.1.2 The extended search space (based on the Spy technique)

The extended search space, proposed in our work [30], introduces three new hyperparameters based on the spy technique for PU learning [8]. Spy-based approaches are used to heuristically determine the $Threshold\_1A$ parameter. A percentage of labelled positive instances (given by Spy_rate) are hidden in the unlabelled set as "spy instances". A classifier ($Classifier\_1A$) is built, using the labelled positive instances as the positive set and the unlabelled instances with the spy instances as the negative set. The spy instances are then classified, and $Threshold\_1A$ is determined such that a percentage of spy instances (given by $Spy\_tolerance$) have a predicted probability of belonging to the positive class of less than $Threshold\_1A$(e.g., if $Spy\_tolerance = 0.05$, 5% of the spy instances can have a predicted positive-class probability less than $Threshold\_1A$). Note that the $Threshold\_1A$ hyperparameter is thus redundant and its value is not used when building the PU learning model for candidate solutions when Spy_flag = True. However, $Threshold\_1A$ is still needed as a component of a candidate solution during the search performed by the Auto-ML system, since some candidate solutions generated along the search will not use the spy technique (depending on the value of the $Flag\_1B$ hyperparameter).

The three new hyperparameters introduced into the extended search space are:

- Spy_flag: { True, False }
- Spy_rate: { 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35 }
- $Spy\_tolerance$: { 0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 }Spy_flag is a Boolean value used to indicate whether or not to use a spy-based



Phase 1A						Phase 1B			Phase 2
Iteration_count_1A	Threshold_1A	Classifier_1A	**Spy_flag**	**Spy_rate**	**Spy_tolerance**	Threshold_1B	Classifier_1B	Flag_1B	Classifier_2

Phase 1A						Phase 1B			Phase 2
5	0.25	Decision Tree	**TRUE**	**0.2**	**0.01**	0.3	SVM	TRUE	Deep Forest

Fig. 2 Example of a candidate solution in the extended search space, with the additional hyperparameters highlighted bold

method in Phase 1A. Spy_rate specifies the percentage of positive instances to use as spies. $Spy\_tolerance$ determines what percentage of spies can remain in the unlabelled set when $Threshold\_1A$ threshold is calculated. The inclusion of these three new hyperparameters increases the size of the size space by a factor of 154 ($2 \times 7 \times 11$) with respect to the size of the base search space, so the size of the extended search space is 1,796,256,000 candidate solutions.

The motivation for expanding the search space was to attempt to increase predictive performance of the system by utilising spy-based methods, as initially proposed in [8]. This approach has been used frequently in the PU learning literature with success [13–19].

Figure 2 shows the representation of a candidate solution in the extended search space, including an example of the value taken by each hyperparameter (specifying an example candidate solution). As the value of Spy_flag is set to True, 20% of the labelled positive instances (determined by Spy_rate) are hidden in the unlabelled set in Phase 1A. The RN threshold is determined as the value at which only 1% of spy instances have a predicted positive-class probability less than the determined value.

Spies are utilised in Phase 1A, but not in Phase 1B. This decision was made as preliminary experiments showed no increase in predictive performance when the system allowed spies in Phase 1B. Also, the search space would be greatly expanded if spies were used in Phase 1B, as the three new hyperparameters introduced in this expanded search space would all need to be repeated for Phase 1B, further increasing the size of the search space by another factor of 154. Thus, if spies were used in Phase 1B, the size of the search space would be calculated as $1{,}796{,}256{,}000 \times 154$, i.e., 276,623,424,000 candidate solutions.

Hence, the use of spies in Phase 1B was not cost-effective in our preliminary experiments, and so we included the spy-based heuristic method in Phase 1A only.

3.2 Candidate solution assessment

The objective function is used to assess the quality of a given configuration of PU learning hyperparameter settings for a specific PU learning task, i.e., a specific input dataset. This is done by applying the PU method configuration defined by the candidate solution to the training set. This assessment procedure was described in detail in our previous work [11, 30], but is briefly outlined here for the reader's convenience.

The Auto-PU systems use an internal cross-validation on the training set in order to assess the quality of candidate solutions. The training set is split into 5 folds, and each candidate PU learning algorithm is run 5 times, each time using a different fold as the validation set and the other 4 folds as the learning set. Each time, a candidate two-step PU learning algorithm (specified by the configuration of the candidate solution) is trained on the learning set, and the F-measure of the learned model is assessed on the validation set. That candidate algorithm's quality is the mean F-measure over the 5 validation sets.

The formulas for precision, recall, and F-measure are given in (1), (2) and (3).

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{1}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2}$$

$$\text{F-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

Where True Positives are the number of positive instances correctly predicted as belonging to the positive class, False Positives are the number of negative instances incorrectly predicted as belonging to the positive class, and False Negatives are the number of positive instances incorrectly predicted as belonging to the negative class.

4 Three Auto-ML Systems for Positive-Unlabelled Learning (Auto-PU)

This section describes the three Auto-PU systems, namely GA-Auto-PU, BO-Auto-PU, and EBO-Auto-PU. Whilst each system utilises a different optimisation method (Genetic Algorithms, Bayesian Optimisation and hybrid Evolutionary/Bayesian Optimisation), all use the same search spaces and objective function, defined in Section 3. Open source implementations of the systems and the baseline methods are available on GitHub[1].

[1] https://github.com/jds39/GA-Auto-PU/tree/main

4.1 GA-Auto-PU

The GA-Auto-PU system has been described in detail in [11, 30]. Hence, we provide only a brief overview of the system here, for the reader's convenience.

GA-Auto-PU is a Genetic Algorithm (GA)-based Auto-ML system for PU learning. For an overview of GAs, see Section 2.1. Algorithm 1 outlines the GA procedure. Initially, a Population of Pop_size individuals is generated (step 1). The configuration (genome) of the individual is checked against a list of previously assessed configurations, and if it has not already been assessed, the *Fitness* (predictive accuracy) of *Individual* is calculated (step 2.a.i.1) as described in Section 3.2. If the configuration has already been assessed, the fitness values of the previous assessment are assigned to the individual (step 2.a.i.2). Once all individuals have been evaluated, the fittest *Individual* is saved for the next generation (step 2.b). A new population is created from *Population* after undergoing tournament selection based on fitness (step 2.c), and New_pop then undergoes crossover (step 2.d) and mutation (step 2.e) to create new candidate solutions. Finally, *Population* is set as New_pop and $Fittest\_individual$(step 2.f). This process of fitness calculation, selection, crossover, mutation, and elitism is repeated #generations times. The fitness of an individual is the F-measure achieved over the internal 5-fold cross-validation on the training set.

Algorithm 1 Outline of the GA procedure

```
1. $Population$ = Generate population();
2. for $\#generations$ times do
    2.a. for each $Individual$ in $Population$ do
        2.a.i. if $Individual$ configuration has not already been assessed then
            2.a.i.1. Assess fitness($Individual$, $Training\ set$);
        2.a.i. else
            2.a.i.2. $IndividualFitness$ value is set as the output of the previous assessment;
    2.b. $Fittest\_individual$ = Get fittest individual($Population$);
    2.c. $New\_pop$ = 100 individuals selected from $Population$ by tournament selection;
    2.d. Individuals in $New\_pop$ undergo crossover;
    2.e. Individuals in $New\_pop$ undergo mutation;
    2.f. $Population = New\_pop \cup \{Fittest\_individual\}$;
3. Return $Individual$ in $Population$ with the highest fitness;
```

The system utilises standard uniform crossover and mutation (replacing a gene's value by a randomly sampled candidate value) as search operators.

4.2 BO-Auto-PU

In this section we introduce BO-Auto-PU, a new Bayesian Optimisation (BO)-based Auto-ML system for PU learning. GA-Auto-PU, described in Section 4.1, was the first Auto-ML system specific to PU learning, and showed statistically significant improvements in predictive accuracy over PU learning baselines [11, 30]. However, GA-Auto-PU is computationally expensive, averaging about 225 minutes to run 5-fold cross-validation per dataset [11, 30].

BO is generally a much more computationally efficient procedure than a standard GA, given that it assesses most of the generated individuals (candidate solutions) with a fast executed surrogate model as opposed to the slowly executed objective function. As such, BO-Auto-PU has been developed in an attempt to reduce the computational expense of GA-Auto-PU without sacrificing predictive accuracy. Algorithm 2 outlines the BO procedure.

Initially, $\#Configs$ PU learning algorithm configurations are randomly generated (step 1) and evaluated, with their F-measures saved as $Scores$(step 2). A random forest regressor, $Surr\_model$(surrogate model), is then trained using *Configs* as features and Scores as the target variable (step 3). In each iteration, a new set of $\#Configs$ configurations, $New\_configs$, are randomly generated (step 4.a) and a surrogate score for each is calculated by $Surr\_model$ and saved as $\hat{Y}$(step 4.b). The best configuration, $Best\_config$, with the highest score according to $\hat{Y}$ is evaluated using the objective function (steps 4.c,d), and added to *Configs*, with the objective $Score$(F-measure) added to *Scores* (step 4.e). $Surr\_model$ is then retrained with $Best\_config$ added to *Config* (step 4.f). Steps 4.a-f are repeated It_count times. Finally, the configuration with the best objective score is returned.

Configs are processed as follows for training $Surr\_model$: for the base search space (without the Spy method), the numeric components of each configuration ($Threshold\_1A$, $Iteration\_Count\_1A$, $Threshold\_1B$) are treated as numeric features, the Boolean component ($Flag\_1B$) is treated as a binary feature, and the nominal components ($Classifier\_1A$, $Classifier\_1B, Classifier\_2$) are one-hot encoded, with a binary value for each potential value of the component, indicating whether or not that value is used. The resulting instance for the regression algorithm (which will learn $Surr\_model$) has 58 features. For the extended search space (with the Spy method), all the previously mentioned components are treated as they are in the base search space. However, we also have the Spy method's components, with Spy_rate and $Spy\_tolerance$ treated as numeric features, and the Boolean component "Spies" treated as a binary feature– a flag, indicating whether or not the Spy method is used. This results in an instance with 61 features for the regression algorithm.

Algorithm 2 Outline of the BO procedure for Positive-Unlabelled learning

```
1. $Configs$ = randomly generate $\#Configs$ PU learning configurations;
2. $Scores$ = run objective function for all configurations in $Configs$;
3. Fit $Surr\_model$ with $Configs$ as features and $Scores$ as the target;
4. **for** $\#It\_count$ times **do**
    4.a. $New\_configs$ = randomly generate $\#Configs$ configurations;
    4.b. $\hat{Y}$ = calculate a surrogate score for each new config with $Surr\_model$;
    4.c. $Best\_config$ = config with the highest score according to $\hat{Y}$;
    4.d. $Score$ = run the objective function for $Best\_config$;
    4.e. Add $Best\_config$ to $Configs$, add $Score$ to $Scores$;
    4.f. Retrain $Surr\_model$ on $Configs$ and $Scores$;
5. **Return** the Best configuration according to the objective score;
```

Table 1 shows the hyperparameter settings of the BO underlying BO-Auto-PU. The It_count parameter determines the number of iterations to perform the optimisation. $\#Configs$ determines the number of individuals in the population. $Surr\_model$ is the surrogate model used to calculate the surrogate score. The acquisition function is the method for selecting which candidate solution to assess with the objective function. For this, we simply use the surrogate score calculated by $Surr\_model$. The It_count and $\#Configs$ parameters were set to match the $\#generations$ and Pop_size parameters for GA-Auto-PU. In Bayesian optimisation, it is common to use an acquisition function that attempts to trade-off exploration and exploitation. However, in our case, preliminary experiments showed worse performance utilising these acquisition functions than simply utilising the surrogate score. As such, we have used the surrogate score only.

4.3 EBO-Auto-PU

As mentioned earlier, GA-Auto-PU's performance came at a large computational cost, averaging about 225 minutes for a 5-fold cross-validation per dataset. In an effort to reduce the computational cost of GA-Auto-PU, we introduced BO-Auto-PU, detailed in the previous section. BO-Auto-PU proved much more computationally efficient than GA-Auto-PU, with BO-Auto-PU averaging less than 10 minutes to run a 5-fold cross-validation per dataset.

However, the improvement in computational efficiency obtained by BO-Auto-PU is potentially associated with a relative reduction in the ability to perform a global exploration of the search space, by comparison with GA-Auto-PU, as follows.

GA-Auto-PU has large population diversity as it assesses many configurations in each iteration according to the slow objective function, and it creates diversity through crossover and mutation applied to configurations selected based on their fitness (predictive accuracy). Whereas BO-Auto-PU assesses only a single configuration at each iteration according to the slow objective function, namely a configuration that is selected according to the predictive accuracy value predicted by the random forest regressor. At each iteration, a population is randomly generated to be evaluated by the faster surrogate model, rather than evolved to be evaluated by the slower objective (fitness) function as in the case of GA-Auto-PU. So, whilst diversity may occur in the BO search, there is no guarantee that it will benefit the system as the diverse configurations may not be selected for assessment by the objective function.

EBO-Auto-PU was developed in an attempt to bridge this gap between the GA- and the BO-based systems, introducing diversity into BO by evolving a population rather than random population generation, whilst utilising a surrogate model to reduce computational expense and prioritise candidate solutions for assessment according to the objective function. The EBO procedure is akin to a surrogate-assisted genetic algorithm (SAGA), with the difference being the number of candidate solutions selected for evaluation (1 + k, as opposed to the normal 1 in a SAGA) and the manner of selection of the k_pop sample, selected with tournament selection and with genetic operators applied to.

Algorithm 3 outlines the procedure that the new EBO component of EBO-Auto-PU follows to evolve a PU learning algorithm. $\#Configs$ PU learning configurations are randomly generated (step 1) and evaluated, with their F-measures saved as Scores (step 2). A random forest regressor, $Surr\_model$, is then trained, using *Configs* as features, and *Scores* as the target variable (step 3). The configurations in *Configs* are copied to a temporary store $Temp\_Configs$ to undergo crossover and mutation to produce an evolved population of configurations (step 4.a). The surrogate scores of each just-produced configuration in *Configs* is calculated by $Surr\_model$ and saved as $\hat{Y}$(step 4.b), with the configuration with the highest surrogate score saved as $Best\_config$(step 4.c). Tournament selection is utilised to select k candidate solutions according to their objective score (F-measure), which are then added to a population k_pop(step 4.d-e). k_pop then undergoes crossover and mutation to produce a newly evolved population (step 4.f). $Best\_config$ and the configurations from k_pop are assessed according to the objective function to calculate their objective scores, before the configurations are added

Table 1 Hyperparameters of the BO-Auto-PU system, with their default values

Hyperparameter	Value
It_count	50
$\#Configs$	101
$Surr\_model$	Random Forest Regressor
$Acquisition function$	$Surr\_model$ predicted value

to *Configs* and used to update the $Surr\_model$(steps 4.g-i). The objective function cited in step 4.g is defined in Section 3.2. This process (step 4 in Algorithm 3) is repeated It_count times. Finally, the best configuration, according to the objective score, is returned.

Note that the computation of the objective score is much more expensive than the computation of the surrogate score, and hence, at each iteration, just k + 1 configurations ($Best\_config$ and the k configurations in k_pop) have their objective score computed.

Configs are processed as follows for training $Surr\_model$: for the base search space, the numeric components of each configuration ($Threshold\_1A$, $Iteration\_Count\_1A$, $Threshold\_1B$) are treated as numeric features, the Boolean component ($Flag\_1B$) is treated as a binary feature, and the nominal components ($Classifier\_1A$, $Classifier\_1B$, $Classifier\_2$) are one-hot encoded, with a binary value for each potential value of the component, indicating whether or not that value is used by the PU learning method. The resulting instances used as input by the regression algorithm consist of 58 features. For the extended search space, all the previously mentioned components are treated as they are in the base search space. However, we also have the additional spy components, with Spy_rate and $Spy\_tolerance$ treated as numeric features, and the Boolean component "Spies" treated as a binary feature. This results in instances consisting of 61 features.

EBO-Auto-PU's hyperparameters and their candidate values are shown in Table 2. Most of these hyperparameters are inherited from GA-Auto-PU and BO-Auto-PU, as follows.

The hyperparameters $\#Configs$, It_count and $Surr\_model$ in Table 2 are the same as the corresponding hyperparameters in Table 1 for the BO procedure, and these hyperparameters take the same settings in both tables (for both EBO and BO), in order to make the comparison between EBO-Auto-PU and BO-Auto-PU as fair as possible. The hyperparameters "Crossover probability", "Component crossover probability", "Mutation probability", and "Tournament size" in Table 2 are also the same as the hyperparameters "$Cross\_prob$", "$Gene\_cross\_prob$", "$Mutation\_prob$", and "$Tournament\_size$" for the GA-Auto-PU system, respectively; and again, these hyperparameters take the same settings for a fair comparison between EBO-Auto-PU and GA-Auto-PU. The only parameter unique to EBO-Auto-PU is the k parameter, used to determine the number of candidate solutions to be selected with tournament selection to assess according to their objective score.

Table 2 Hyperparameters of the EBO-Auto-PU system, with their default values

Hyperparameter	Value
$\#Configs$	101
It_count	50
$Surr\_model$	Random Forest Regressor
Crossover probability	0.9
Component crossover probability	0.5
Mutation probability	0.1
Tournament size	2
k	10

Algorithm 3 Outline of the EBO procedure for Positive-Unlabelled learning

```
1. $Configs$ = randomly generate $\#Configs$ PU learning configurations;
2. $Scores$ = run objective function for all configurations in $Configs$;
3. Fit $Surr\_model$ with $Configs$ as features and $Scores$ as the target;
4. for $\#It\_count$ times do
    4.a. $Temp\_Configs$ = $Configs$ after undergoing crossover and mutation;
    4.b. $\hat{Y}$ = calculate a surrogate score for each new config in $Temp\_Configs$ with $Surr\_model$;
    4.c. $Best\_config$ = config with the highest score according to $\hat{Y}$;
    4.d. $k\_pop$ = { };
    4.e. for $k$ times do
        4.e.i. $k\_cand\_sol$ = select from $Temp\_Configs$ using tournament selection based on surrogate scores;
        4.e.ii. $k\_pop = k\_pop \cup k\_cand\_sol$;
    4.f. $k\_pop$ = configurations in $k\_pop$ undergo crossover and mutation;
    4.g. Assess $Best\_config$ and each configuration from $k\_pop$ with objective function to obtain objective scores;
    4.h. $Configs = Configs \cup Best\_config \cup k\_pop$;
    4.i. Retrain $Surr\_model$ on $Configs$;
5. Return Best configuration according to the objective score;
```

4.4 Computational efficiency

In the GA-based, BO-based, and EBO-based Auto-ML systems for PU learning, the running time is by far dominated by the time to evaluate the candidate solutions along the iterations of the search, i.e., the time to learn a PU model and evaluate its F-measure on the training set, for each candidate PU learning method. GA, BO, and the EBO-based methods perform the same number of iterations (50) in our experiments. However, in each generation (iteration) of GA-Auto-PU the GA must learn n PU models, where n is the number of individuals (candidate solutions) in the population; each iteration of BO-Auto-PU needs to learn a single PU model; whilst each iteration of EBO-Auto-PU needs to learn k+1 models, where k is the number of candidate solutions selected via tournament selection to assess according to the objective function. Learning a PU model can be very computationally expensive, depending not only on the size of the dataset but also on the time complexity of the 3 classification algorithms chosen for Phases 1A, 1B, and 2 of the 2-step method, and the number of iterations the classifier is applied in Phase 1A.

All three Auto-ML systems also must perform other steps for generating candidate solutions to be evaluated, but these take in general much less time than the time to evaluate candidate solutions using the objective function (F-measure) as described above. More precisely, at each iteration the GA and the EBO must perform tournament selection, crossover and mutation, but these are all simple operations, which are much faster than computing the fitness function (learning one PU model for each individual).

Unlike the GA, at each iteration BO and EBO learn a surrogate model, but again, the time for this is much shorter than the time taken to learn a PU model in each iteration of BO. This is because the surrogate model is learned by a relatively fast random forest algorithm using a small dataset of PU algorithm configurations, whilst learning a PU model involves running multiple classifiers (one of them for several iterations in phase I-A), each classifier can be much slower than a random forest. In addition, each classifier is learned using the training data of the current dataset, which is typically much larger in number of instances than the small dataset of PU method configurations. As EBO-Auto-PU assesses (using the expensive objective function) fewer candidate solutions than GA-Auto-PU, but more than BO-Auto-PU, the EBO-Auto-PU system sits between the GA-Auto-PU and the BO-Auto-PU systems in regard to computational runtime. More precise results about the differences of computational time among these three systems will be reported in Section 6.

Table 3 Main characteristics of the biomedical datasets used in the experiments

Dataset	No. inst.	No. feat.	Posit.-class %
Alzheimer's [36]	354	9	10.73%
Autism [35]	288	15	48.26%
Breast cancer Coimbra [35]	116	9	55.17%
Breast cancer Wisconsin [35]	569	30	37.26%
Breast cancer mutations [37]	1416	53	32.42%
Cervical cancer [35]	668	30	2.54%
Cirrhosis [38]	277	17	25.72%
Dermatology [35]	359	34	13.41%
Pima Indians Diabetes [35]	769	8	34.90%
Early Stage Diabetes [39]	521	17	61.54%
Heart Disease [35]	304	13	54.46%
Heart Failure [40]	300	12	32.11%
Hepatitis C [35]	590	13	9.51%
Kidney Disease [35]	159	24	27.22%
Liver Disease [35]	580	11	71.50%
Maternal Risk [35]	1014	6	26.82%
Parkinsons [35]	196	22	75.38%
Parkinsons Biomarkers [41]	131	29	23.08%
Spine [35]	311	6	48.39%
Stroke [42]	3427	15	5.25%

5 Experimental methodology

5.1 Datasets used in the experiments

Experiments were conducted on 20 publicly available biomedical datasets, including 13 classical benchmark classification datasets from the well-known UCI dataset repository [35], and 7 datasets from [36–42]. These datasets all involve real-world learning scenarios of disease or health-risk prediction. The main characteristics of these datasets (numbers of instances and features, and percentage of positive instances) are shown in Table 3.

These datasets are originally binary classification datasets and so need to be adapted for PU learning. To do so, in each training set, we have hidden δ% of the positive instances in the negative set (where δ is a user-specified parameter), thus creating an unlabelled set. This process of engineering a PU dataset from a binary dataset is common throughout the PU learning literature [8, 23, 43, 44]. δ takes the values 20%, 40%, and 60% throughout this work, meaning that each dataset is engineered into three datasets, thus creating 60 datasets total. Due to the nature of PU learning, it is often hard to know the distribution of positive instances, and what proportion of them remain unlabelled. Hence, to further analyse the performance of a PU learning algorithm, it is important, when feasible, to assess the performance of its learned model on multiple distributions of unlabelled instances. That is, testing on different versions of the same dataset with differing percentages of the positive instances hidden in the unlabelled set in the training set, when doing experiments with engineered PU datasets.

Biomedical datasets are good candidates for PU learning given the inherent uncertainty involved in labelling biomedical data. For example, when learning whether or not a person has a given disease (the scenario for many of our datasets), a classifier may learn to distinguish between a positive set (patients diagnosed with that disease) and a negative set (patients not diagnosed with that disease). From this data, we wish to identify whether an unseen patient *has* a specific disease. However, consider the wording of this scenario. We are looking to identify whether a patient has a disease, by learning from data consisting of patients who have or have not been diagnosed with a disease. In other words, we are looking to identify true positives by learning only from labelled positives. The apparently negative set, in this scenario, can be more precisely considered an unlabelled set, given that "not diagnosed" does not mean that a patient does not have a disease. It might simply be that this patient has not undergone any tests to determine whether the disease is present. Or, this patient may have undergone some tests, but the disease may be undetectable with the given test. For examples of studies detailing the reliability of specific diagnostic tests see [45–50]. Furthermore, biomedical tests are expensive, and thus the presence of unlabelled data may simply be a practicality to minimise the cost of data curation. Thus, we have used biomedical datasets in our experiments as they are appropriate for PU learning and have been referred to as "one of the most significant usage areas in PU learning" [51]. The datasets used in this work are available on GitHub[2].

5.2 Nested cross-validation

Throughout this work, the experiments use a nested cross-validation procedure, with an external cross-validation used to measure predictive performance and an internal cross-validation used to evaluate candidate solutions during a run of an Auto-PU system.

For the external cross-validation, the experiments use the well-known stratified 5-fold cross-validation procedure. This involves randomly splitting the data into 5 folds, and then using one fold at a time as the test set and the other four folds as the training set. The predictive performance of a method is its average performance over the 5 test sets in this external cross-validation. The cross-validation is stratified in the sense that in each of the 5 folds the distribution of class labels is approximately the same as the distribution in the full dataset. We chose 5 folds, rather than the more popular 10 folds, as the number of positive instances in some of our datasets are small. Thus, in some datasets 10 folds would split the positive set into folds that are so small as to be practically unsuitable.

For each iteration of the external cross-validation (i.e. for each pair of training and test sets), we run a stratified internal cross-validation, where the training set is randomly split into 5 folds and the Auto-PU system is run 5 times, using one fold at a time as the validation set (to measure performance) and the other 4 folds as the learning set (to learn a classifier using a candidate PU learning algorithm, out of the algorithms in the Auto-PU system's search space). The performance of a candidate PU learning algorithm is determined by the average F-measure value achieved over the 5 validation sets. When these 5 runs of the Auto-PU system are over, the best PU learning algorithm found by the system is thus the one with the highest average F-measure over the 5 validation sets of the current training set. Then, a PU learning classifier is built from the full training set with the configuration defined by that best PU learning algorithm. The classifier is then used to predict the class of all instances in the test set of the current iteration of the external cross-validation. This process is repeated for the 5 pairs of training and test sets in the external 5-fold cross-validation.

It is important to note that neither the external test sets nor any negative labels are accessed during the hyperparameter tuning process. All evaluation and selection of PU learning configurations within the internal cross-validation is conducted exclusively using positive and unlabelled data from the training folds. Therefore, the integrity of PU learning is maintained and the tuning process does not introduce bias through access to negative examples.

When comparing a version of the Auto-PU system against another version or a PU learning method, all systems/methods tested use the same nested cross-validation procedure, with the same folds, to ensure a fair comparison.

5.3 Statistical significance analysis

We performed two different but related types of statistical analyses, as follows. First, in Sections 6.1 and 6.2, we analyse the results for each search space (base and extended spaces) separately. For this analysis, for each performance measure (F-measure, recall, and precision), we compare the performance of an Auto-ML system against the performance of other Auto-ML systems or baseline PU learning methods using the non-parametric Wilcoxon Signed-Rank test [52]. Since this involved testing multiple null hypotheses, we use the well-known Holm correction [53] for multiple hypothesis testing. This procedure involves comparing the best method against each of the other methods, ranking the p-values from the smallest to largest (i.e., from most to least significant), and adjusting the significance level α according to the p-values' ranking. We set $\alpha = 0.05$ as usual before adjusting it according to the position of the p-value in the ranked list. p_1(the smallest p-value) is deemed significant if it is less than α/n, where n is the number of hypotheses tested, which is the number of pairs of methods being compared. If this condition is not satisfied, the procedure stops and all p-values are deemed nonsignificant. If p_1 is deemed significant, p_2 is deemed significant if it is less than $\alpha/(n-1)$, etc.

As a second type of statistical analysis, in Section 6.3 we compare all Auto-PU systems (for both search spaces) and the two baseline methods as a whole. Due to the larger number of methods being compared in a single statistical analysis, we first apply the non-parametric Friedman test [53] to assess if there is a significant difference among the results of all methods as a whole. If the resulting p-value is significant (smaller than $\alpha = 0.05$), we proceed by applying the Wilcoxon Signed-Rank test with Holm correction to each pair of methods, as described above.

[2] https://github.com/jds39/Unlabelled-Datasets

5.4 Hyperparameter optimisation for the baseline PU learning methods

Since the Auto-PU systems optimise the configuration of a PU learning algorithm, we also optimised the hyperparameters of the baseline PU learning methods (S-EM, DF-PU) with a grid search (via an internal cross-validation on the training set), for a fair comparison. For DF-PU, we optimised the percentage of unlabelled instances selected as reliable negatives (reliable negative rate) and the number of iterations, with the following candidate values:

- Reliable negative rate: { 0.01, 0.03, 0.05, 0.07, 0.09, 0.11, 0.13, 0.15, 0.17, 0.2 }
- Iteration count: { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }

For S-EM, the hyperparamters optimised were the same as those optimised for the extended search space of the Auto-PU systems; namely the spy rate and the spy tolerance. The candidate values of these hyperparameters were as follows.

- Spy rate: { 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35 }
- Spy tolerance: { 0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 }

6 Results and discussion

This section consists of three parts. Section 6.1 reports the results for the base search space. Section 6.2 reports the results for the extended search space. Each of these two sections has two parts, first comparing the Auto-PU systems among themselves and then comparing the Auto-PU systems against two baseline PU learning methods, namely S-EM [8] and DF-PU [12]. Section 6.3 compares all methods (Auto-PU systems for both search spaces and baseline methods) as a whole. This Section 6 reports the F-measure results per dataset, as this is the most popular metric in PU learning [23]. The precision and recall results per dataset are not reported to save space; but this section reports statistical significance results for all three measures (F-measure, precision, recall), for all datasets as a whole.

Hereafter we will use the short names GA-1, BO-1 and EBO-1 to refer to the versions of GA-Auto-PU, BO-Auto-PU and EBO-Auto-PU with the *base* search space; and use the short names GA-2, BO-2 and EBO-2 to refer to the versions of GA-Auto-PU, BO-Auto-PU and EBO-Auto-PU with the *extended* search space. We will also use the term Auto-PU to refer to the Auto-ML systems for PU learning.

6.1 Results for the three Auto-PU systems with the base search space

Recall that the base search space allows for simple PU learning algorithms to be created without spy-based methods (which are included in the extended search space).

6.1.1 Comparing the three Auto-PU systems

Table 4 reports the F-measure values achieved by the three Auto-PU systems with the base search space on each of the 20 datasets, for δ = 20%, 40%, 60%, along with the average F-measure for each Auto-PU system across the datasets for each δ value. The final column shows the percentage of positive instances in the dataset. Recall that δ denotes the percentage of positive instances hidden in the unlabelled set.

Table 5 reports the statistical significance results for the comparison of the Auto-PU systems in terms of F-measure, precision and recall. For each combination of a performance measure (F-measure, precision, recall) and a δ value (δ= 20%, 40%, 60%), this table has one row for each pair of Auto-PU systems being compared, and for each such pairwise comparison, the table reports the average (avg.) rank of each of the two methods being compared and the corresponding p-value and adjusted α value (significance level), based on the Holm correction [53]. The better (lower) avg. rank in each cell is shown in boldface. For example, in the cell for F-measure, δ = 20%, Compared Systems GA-1 vs BO-1, the average ranks for those two systems are 1.52 and 1.48, respectively. Hence, BO-1 was the winner, but the p-value (0.952) was greater than the adjusted α, so this result was not statistically significant. The following discussion of results will focus mainly on the F-measure, the most important measure in Table 5, as mentioned earlier.

Table 5 shows little difference in performance between the methods for δ = 20%. For the F-measure results, EBO-1 ties with GA-1 and slightly outperforms BO-1; and BO-1 slightly outperformed GA-1. The results for δ = 40% show EBO-1 slightly outperforming both GA-1 and BO-1, whilst BO-1 outperforms GA-1. For δ = 60%, F-measure results, GA-1 substantially outperformed BO-1 (avg. rank **1.25** vs 1.75) and slightly outperformed EBO-1; whilst EBO-1 slightly outperformed BO-1. However, there are no statistically significant results for any δ value in this table.

Based on these results, it can be argued that overall EBO-1 performed best for δ = 20% and δ = 40%, whilst GA-1 performed best for δ = 60%; but no statistical significance was achieved in any experiment, and the performance gains of EBO-1 and GA-1 were in general marginal– except that GA-1 had a substantially better avg. rank than BO-1 for δ = 60%.

Table 4 F-measure results of the Auto-PU systems utilising the base search space

Dataset	δ=20%			δ=40%			δ=60%			% Pos
	EBO-1	GA-1	BO-1	EBO-1	GA-1	BO-1	EBO-1	GA-1	BO-1	
Alzheimer's	0.629	0.529	0.615	0.587	0.551	0.600	0.540	0.456	0.436	10.73
Autism	0.986	0.960	0.967	0.926	0.927	0.956	0.927	0.910	0.863	48.26
Breast cancer Coi.	0.966	0.705	0.694	0.952	0.687	0.701	0.615	0.510	0.586	55.17
Breast cancer Wis.	0.893	0.954	0.949	0.872	0.932	0.969	0.927	0.906	0.895	37.26
Breast cancer mut.	0.672	0.893	0.893	0.667	0.868	0.873	0.862	0.854	0.841	32.42
Cervical cancer	0.839	0.828	0.839	0.904	0.903	0.903	0.667	0.714	0.645	2.54
Cirrhosis	0.532	0.573	0.545	0.453	0.464	0.529	0.507	0.443	0.489	25.72
Dermatology	0.899	0.860	0.872	0.813	0.780	0.905	0.716	0.828	0.725	13.41
PI Diabetes	0.654	0.677	0.647	0.661	0.649	0.645	0.634	0.606	0.594	34.90
ES Diabetes	0.973	0.958	0.983	0.913	0.895	0.877	0.909	0.930	0.902	61.54
Heart Disease	0.833	0.843	0.844	0.800	0.801	0.830	0.774	0.785	0.777	54.46
Heart Failure	0.732	0.770	0.753	0.666	0.652	0.605	0.640	0.674	0.704	32.11
Hepatitis C	0.925	0.953	0.907	0.835	0.771	0.838	0.667	0.588	0.708	9.51
Kidney Disease	1.000	0.976	0.988	0.938	0.988	0.964	0.646	0.754	0.806	27.22
Liver Disease	0.827	0.834	0.820	0.819	0.803	0.817	0.717	0.804	0.795	71.50
Maternal Risk	0.855	0.476	0.837	0.803	0.812	0.780	0.739	0.735	0.689	26.82
Parkinsons	0.929	0.860	0.935	0.894	0.836	0.875	0.707	0.818	0.732	75.38
Parkinsons Biom.	0.203	0.476	0.167	0.337	0.265	0.192	0.133	0.233	0.182	23.08
Spine	0.933	0.652	0.954	0.932	0.907	0.926	0.775	0.818	0.742	48.39
Stroke	0.239	0.474	0.244	0.225	0.255	0.153	0.229	0.255	0.208	5.25
Average	0.776	0.763	0.773	0.750	0.737	0.747	0.667	0.681	0.666	–

Table 5 Results of Wilcoxon signed-rank tests with Holm correction for multiple hypothesis when comparing each pair of Auto-PU systems (with the base search space) regarding F-measure, Precision and Recall, for the three δ values

δ(%)	Compared systems	F-measure			Precision			Recall		
		Avg ranks	p-value	α	Avg ranks	p-value	α	Avg ranks	p-value	α
20%	GA-1 vs BO-1	1.52 vs **1.48**	0.952	0.05	1.55 vs **1.45**	0.983	0.05	**1.4** vs 1.6	0.114	0.017
	GA-1 vs EBO-1	1.5 vs 1.5	0.784	0.025	1.52 vs **1.48**	0.446	0.017	**1.38** vs 1.62	0.178	0.025
	BO-1 vs EBO-1	1.52 vs **1.48**	0.658	0.017	1.55 vs **1.45**	0.968	0.025	1.55 vs **1.45**	0.983	0.05
40%	GA-1 vs BO-1	1.62 vs **1.38**	0.334	0.025	1.55 vs **1.45**	0.601	0.025	1.58 vs **1.42**	0.398	0.025
	GA-1 vs EBO-1	1.6 vs **1.4**	0.245	0.017	**1.45** vs 1.55	0.828	0.05	1.58 vs **1.42**	0.381	0.017
	BO-1 vs EBO-1	1.55 vs **1.45**	1.000	0.05	**1.45** vs 1.55	0.387	0.017	1.5 vs 1.5	0.557	0.05
60%	GA-1 vs BO-1	**1.25** vs 1.75	0.177	0.017	**1.48** vs 1.52	0.513	0.025	**1.25** vs 1.75	0.064	0.017
	GA-1 vs EBO-1	**1.45** vs 1.55	0.312	0.025	1.5 vs 1.5	0.812	0.05	**1.45** vs 1.55	0.388	0.025
	BO-1 vs EBO-1	1.6 vs **1.4**	0.674	0.05	**1.38** vs 1.62	0.184	0.017	1.62 vs **1.38**	0.629	0.05

Figure 3 shows how the F-measure values achieved by the three Auto-PU systems and the two baseline PU learning methods (DF-PU and S-EM) vary across the three δ values. The results of the baseline methods will be discussed in the next Subsection.

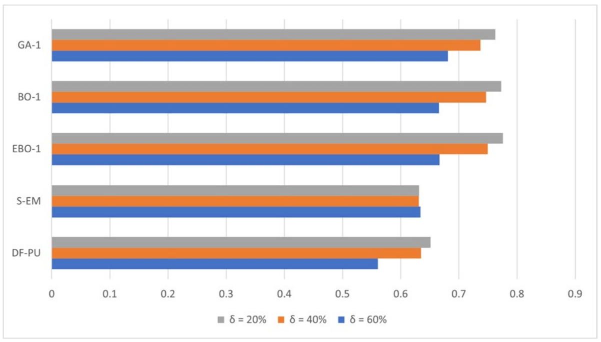


Fig. 3 Average F-measure results comparison for three versions of Auto-PU utilising the base search space and the PU learning baselines across the three values of δ

Regarding the results for GA-1, BO-1 and EBO-1 in this figure, their F-measure values decrease with an increase in the δ value, i.e. they all achieved their best and their worst F-measure values when $\delta = 20\%$ and $\delta = 60\%$, respectively. This is not surprising, since δ represents the percentage of positive instances hidden in the unlabelled set; and intuitively, the larger this percentage, the more difficult the PU learning problem is. The decrease in F-measure values is relatively small when δ is increased from 20% to 40%, but there is a more substantial decrease in F-measure value when δ is increased to 60%, for all the three Auto-PU systems.

Table 6 shows the values of Pearson's linear correlation coefficient between the F-measure values of each Auto-PU

Table 6 Linear (Pearson's) correlation coefficient value between the F-measure and the percentage of positive examples in the original dataset (before hiding some positive examples in the unlabelled set) for each combination of an Auto-PU system (with base search space) or PU method

Auto-PU system or PU method	δ = 20%	δ = 40%	δ = 60%
EBO-1	0.440	0.469	0.460
BO-1	0.398	0.360	0.498
GA-1	0.333	0.385	0.504
DF-PU	0.381	0.504	0.445
S-EM	0.690	0.631	0.686

system or baseline PU learning method and the percentages of positive examples in the original dataset, for each δ value. For all systems or methods, for all δ values, there is a positive correlation between these two factors. That is, in general, the larger the percentage of positive-class instances, the higher the F-measure value, probably due to less class imbalance in the dataset. To simplify the analysis of these correlations, we categorise the positive correlations as specified in [54]: 0.00–0.10 negligible correlation, 0.10–0.39 weak correlation, 0.40–0.69 moderate correlation, 0.70–0.89 strong correlation, 0.90–1.00 very strong correlation.

Of the three Auto-PU systems, EBO-1 shows the strongest positive correlation overall, achieving a moderate correlation for each δ value. BO-1 and GA-1 both show weak correlations for δ = 20% and δ = 40%, but both show moderate correlation for δ = 60%. Whilst the correlation for EBO-1 is relatively stable across δ values, BO-1 and GA-1 both show a substantial increase for δ = 60%. The correlations for the two baseline PU learning methods will be discussed in the next subsection.

6.1.2 Comparing the Auto-PU Systems against two baseline PU methods

Table 7 reports the F-measure values achieved by the baseline PU learning methods DF-PU and S-EM (see Section 2.1). Table 8 reports the the statistical significance results for F-measure, precision and recall. For each δ value, this table shows the statistical results for 6 pairwise comparisons, by comparing each of the three Auto-PU systems against each of the two baseline PU learning methods. The statistical results for each pairwise comparison follow the same format as described earlier for Table 5. In each comparison, the better (lower) avg. rank in each cell is shown in boldface; and a p-value is shown in boldface if it is significant (i.e., smaller than the adjusted α). For example, in the first row of results in Table 8, for δ= 20%, comparing GA-1 vs DF-PU, regarding F-measure, the average ranks of GA-1 and DF-PU were 1.05 and 1.95 respectively, and this result was statistically significant. The following discussion of results will focus mainly on the F-measure, the most important measure in Table 8, whilst precision and recall results are reported for completeness.

Table 7 F-measure results of the baseline PU learning methods

Dataset	δ=20%		δ=40%		δ=60%		% Pos
	DF-PU	S-EM	DF-PU	S-EM	DF-PU	S-EM	
Alzheimer's	0.519	0.385	0.396	0.489	0.462	0.389	10.73
Autism	0.766	0.823	0.639	0.824	0.557	0.838	48.26
Breast cancer Coi.	0.680	0.896	0.894	0.903	0.618	0.903	55.17
Breast cancer Wis.	0.884	0.892	0.700	0.893	0.473	0.892	37.26
Breast cancer mut.	0.500	0.711	0.500	0.690	0.500	0.702	32.42
Cervical cancer	0.688	0.054	0.667	0.053	0.650	0.046	2.54
Cirrhosis	0.458	0.438	0.389	0.442	0.364	0.465	25.72
Dermatology	0.872	0.718	0.712	0.718	0.860	0.719	13.41
PI Diabetes	0.566	0.536	0.582	0.535	0.584	0.575	34.90
ES Diabetes	0.751	0.796	0.726	0.863	0.700	0.783	61.54
Heart Disease	0.719	0.838	0.701	0.838	0.730	0.828	54.46
Heart Failure	0.590	0.490	0.586	0.514	0.457	0.584	32.11
Hepatitis C	0.916	0.701	0.804	0.708	0.683	0.661	9.51
Kidney Disease	0.874	1.000	1.000	1.000	0.607	0.951	27.22
Liver Disease	0.806	0.834	0.829	0.726	0.753	0.798	71.50
Maternal Risk	0.459	0.457	0.474	0.444	0.418	0.499	26.82
Parkinsons	0.857	0.847	0.857	0.751	0.799	0.767	75.38
Parkinsons Biom.	0.313	0.308	0.375	0.276	0.065	0.331	23.08
Spine	0.651	0.806	0.654	0.852	0.723	0.840	48.39
Stroke	0.155	0.102	0.214	0.102	0.207	0.103	5.25
Average	0.651	0.632	0.635	0.631	0.561	0.634	–

Table 8 Results of Wilcoxon signed-rank tests with Holm correction for multiple hypothesis when comparing each of GA-1, BO-1 and EBO-1 (with the base search space) against each of two baseline PU learning methods (DF-PU and S-EM) regarding F-measure, Precision and Recall, for the three δ values

δ(%)	Compared systems	F-measure			Precision			Recall		
		Ranking	P-value	Adj. α	Ranking	P-value	Adj. α	Ranking	P-value	Adj. α
20%	GA-1 vs DF-PU	**1.05** vs 1.95	**1E-05**	0.025	**1.2** vs 1.8	**0.001**	0.025	**1.38** vs 1.62	0.494	0.05
	GA-1 vs S-EM	**1.15** vs 1.85	**0.006**	0.05	**1.18** vs 1.82	**0.002**	0.05	1.85 vs **1.15**	**0.011**	0.025
	BO-1 vs DF-PU	**1.15** vs 1.85	**3E-04**	0.025	**1.25** vs 1.75	**0.002**	0.025	**1.4** vs 1.6	0.189	0.025
	BO-1 vs S-EM	**1.2** vs 1.8	**0.003**	0.05	**1.22** vs 1.78	**0.005**	0.05	1.68 vs **1.32**	0.266	0.05
	EBO-1 vs DF-PU	**1.05** vs 1.95	**6E-05**	0.025	**1.1** vs 1.9	**1E-05**	0.025	**1.38** vs 1.62	0.494	0.025
	EBO-1 vs S-EM	**1.22** vs 1.78	**0.002**	0.05	**1.08** vs 1.92	**2E-04**	0.05	1.78 vs **1.22**	**0.018**	0.05
40%	GA-1 vs DF-PU	**1.3** vs 1.7	**0.007**	0.05	**1.12** vs 1.88	**0.001**	0.05	1.55 vs **1.45**	**0.841**	0.05
	GA-1 vs S-EM	**1.2** vs 1.8	**0.002**	0.025	**1.12** vs 1.88	**0.002**	0.025	1.75 vs **1.25**	**0.011**	0.025
	BO-1 vs DF-PU	**1.25** vs 1.75	**0.008**	0.05	**1.12** vs 1.88	**0.001**	0.025	1.58 vs **1.42**	0.629	0.05
	BO-1 vs S-EM	**1.2** vs 1.8	**0.003**	0.025	**1.12** vs 1.88	**0.002**	0.05	1.8 vs **1.2**	**0.003**	0.025
	EBO-1 vs DF-PU	**1.15** vs 1.85	**2E-04**	0.025	**1.08** vs 1.92	**2E-04**	0.025	1.5 vs 1.5	0.777	0.05
	EBO-1 vs S-EM	**1.2** vs 1.8	**4E-04**	0.05	**1.08** vs 1.92	**3E-04**	0.05	1.7 vs **1.3**	0.030	0.025
60%	GA-1 vs DF-PU	**1.2** vs 1.8	**0.003**	0.025	**1.1** vs 1.9	**0.001**	0.05	1.55 vs **1.45**	0.571	0.05
	GA-1 vs S-EM	**1.35** vs 1.65	0.216	0.05	**1.22** vs 1.78	**0.001**	0.025	2.0 vs **1.0**	**2E-06**	0.025
	BO-1 vs DF-PU	**1.25** vs 1.75	**0.011**	0.025	**1.1** vs 1.9	**0.001**	0.025	1.52 vs **1.48**	0.748	0.05
	BO-1 vs S-EM	**1.35** vs 1.65	0.409	0.05	**1.12** vs 1.88	**0.001**	0.05	1.9 vs **1.1**	**4E-05**	0.025
	EBO-1 vs DF-PU	**1.25** vs 1.75	**0.006**	0.025	**1.15** vs 1.85	**0.001**	0.05	1.55 vs **1.45**	0.701	0.05
	EBO-1 vs S-EM	**1.4** vs 1.6	0.498	0.05	**1.15** vs 1.85	**1E-04**	0.025	1.9 vs **1.1**	**4E-05**	0.025

Table 9 Average runtime per dataset (in minutes), on a 48-core GPU system, for Auto-PU systems using the base search space and the two baseline PU learning methods

Method	Avg. Runtime (min)
GA-1	226.3
EBO-1	24.2
BO-1	8.4
DF-PU	4.9
S-EM	1.5

Considering the results shown in Table 8, all three Auto-PU systems performed favourably across all comparisons against the baselines (DF-PU and S-EM), outperforming the baselines in every case for F-measure and precision, with statistical significance in the vast majority of cases. For recall, in general, the baseline methods performed better, with statistical significance in several cases.

As shown by Figure 3, although the Auto-PU systems exhibit a significant performance decrease for δ = 60%, S-EM's performance remains consistent throughout. However, the average performance of S-EM is lower than the average performance of all versions of Auto-PU for all δ values. Hence, even the worst average performance of Auto-PU is better than the best average performance of either baseline.

Considering all results from this Section 6.1, we can conclude that the Auto-PU systems outperform the baselines. The only measure on which the Auto-PU systems are outperformed by the baselines is recall, but this is largely due to their overpredictions of the positive class, which raises recall but consequently decreased precision due to the increase in the number of false positive predictions, leading to a reduced F-measure value overall.

However, predictive accuracy is not the only metric to consider when evaluating these Auto-PU systems, especially since none achieved statistically significantly better results than the other two systems. Computational cost is also a factor to consider, due to the very large computational cost generally associated with Auto-ML systems. So, we measured the computational times to run a nested cross-validation per dataset of these systems, running on a 48 core GPU, and reported in Table 9. BO-1 ran about 27 times faster than GA-1, and almost three times as fast as EBO-1. In addition, EBO-1 ran almost 10 times faster than GA-1. So, both BO-1 and EBO-1 represent a very significant improvement over GA-1 in regard to computational cost. As expected DF-PU and S-EM have smaller runtimes than the Auto-PU systems, but this runtime reduction comes with significantly reduced predictive accuracy, a bad trade-off.

In summary, since EBO-1 achieved overall the best F-measure results for the base search space as mentioned earlier, alongside its large improvement in computational time over GA-1, arguably EBO-1 is the best of the three Auto-PU systems.

Regarding the correlation between the percentage of positive instances and F-measure (Table 6), S-EM shows a stronger correlation across all three δ values than the Auto-PU systems. DF-PU exhibits a similar correlation to the Auto-PU systems.

Table 10 F-measure results of the Auto-PU systems utilising the extended search space

Dataset	δ=20%			δ=40%			δ=60%			% Pos
	EBO-2	BO-2	GA-2	EBO-2	BO-2	GA-2	EBO-2	BO-2	GA-2	
Alzheimer's	0.559	0.580	0.548	0.597	0.603	0.576	0.581	0.492	0.529	10.73
Autism	0.964	0.963	0.982	0.938	0.937	0.940	0.887	0.914	0.927	48.26
Breast cancer Coi.	0.967	0.667	0.711	0.952	0.618	0.671	0.923	0.000	0.553	55.17
Breast cancer Wis.	0.882	0.959	0.956	0.863	0.942	0.936	0.839	0.889	0.866	37.26
Breast cancer mut.	0.666	0.890	0.896	0.655	0.853	0.739	0.587	0.845	0.872	32.42
Cervical cancer	0.867	0.867	0.867	0.904	0.867	0.839	0.516	0.839	0.350	2.54
Cirrhosis	0.506	0.497	0.446	0.493	0.515	0.397	0.322	0.472	0.204	25.72
Dermatology	0.857	0.876	0.901	0.891	0.841	0.896	0.750	0.795	0.692	13.41
PI Diabetes	0.668	0.653	0.642	0.665	0.648	0.646	0.607	0.615	0.634	34.90
ES Diabetes	0.957	0.954	0.978	0.905	0.891	0.887	0.915	0.912	0.894	61.54
Heart Disease	0.826	0.844	0.836	0.804	0.817	0.780	0.747	0.805	0.786	54.46
Heart Failure	0.741	0.757	0.751	0.656	0.652	0.670	0.514	0.600	0.671	32.11
Hepatitis C	0.907	0.964	0.944	0.907	0.761	0.863	0.689	0.612	0.610	9.51
Kidney Disease	0.911	0.976	0.925	0.897	0.976	0.951	0.656	0.789	0.806	27.22
Liver Disease	0.832	0.822	0.831	0.800	0.815	0.817	0.748	0.722	0.748	71.50
Maternal Risk	0.854	0.847	0.862	0.810	0.786	0.813	0.731	0.729	0.738	26.82
Parkinsons	0.914	0.936	0.935	0.850	0.837	0.843	0.720	0.800	0.792	75.38
Parkinsons Biom.	0.259	0.286	0.282	0.276	0.000	0.259	0.203	0.000	0.280	23.08
Spine	0.942	0.941	0.923	0.920	0.936	0.917	0.802	0.700	0.761	48.39
Stroke	0.232	0.256	0.241	0.225	0.255	0.239	0.201	0.233	0.243	5.25
Average	0.766	0.777	0.773	0.750	0.728	0.734	0.647	0.638	0.648	–

Table 11 Results of Wilcoxon signed-rank tests with Holm correction for multiple hypothesis when comparing each pair of Auto-PU systems (with the extended search space) regarding F-measure, Precision and Recall, for the three δ values

δ(%)	Compared systems	F-measure			Precision			Recall		
		Avg. rank	p-value	adj. α	Avg. rank	p-value	adj. α	Avg. rank	p-value	adj. α
20%	GA-2 vs BO-2	1.5 vs 1.5	0.514	0.017	1.5 vs 1.5	0.647	0.05	1.58 vs 1.42	0.376	0.017
	GA-2 vs EBO-2	**1.38** vs 1.62	0.658	0.025	**1.45** vs 1.55	0.632	0.025	**1.42** vs 1.58	0.758	0.05
	BO-2 vs EBO-2	1.62 vs **1.38**	0.825	0.05	1.52 vs **1.48**	0.368	0.017	**1.45** vs 1.55	0.396	0.025
40%	GA-2 vs BO-2	1.55 vs **1.45**	0.729	0.05	1.52 vs **1.48**	0.387	0.05	**1.38** vs 1.62	0.520	0.05
	GA-2 vs EBO-2	1.65 vs **1.35**	0.039	0.017	1.7 vs **1.3**	**0.013**	0.017	1.62 vs **1.38**	0.356	0.017
	BO-2 vs EBO-2	1.52 vs **1.48**	0.220	0.025	1.62 vs **1.38**	0.297	0.025	1.52 vs **1.48**	0.376	0.025
60%	GA-2 vs BO-2	**1.4** vs 1.6	0.388	0.017	**1.42** vs 1.58	0.387	0.025	**1.45** vs 1.55	0.784	0.05
	GA-2 vs EBO-2	1.5 vs 1.5	0.756	0.025	1.52 vs **1.48**	0.702	0.05	**1.45** vs 1.55	0.622	0.025
	BO-2 vs EBO-2	**1.4** vs 1.6	0.956	0.05	1.6 vs **1.4**	0.231	0.017	**1.42** vs 1.58	0.587	0.017

6.2 Results for the Three Auto-PU systems with the extended search space

Recall that the extended search space allows for PU learning algorithms to be created utilising the spy approach (unlike the base search space).

6.2.1 Comparing the three Auto-PU systems

Table 10 reports the F-measure values achieved by the three Auto-PU systems with the extended search space on each of the 20 datasets, for δ = 20%, 40%, 60%; whilst Table 11 reports the statistical significance results comparing the Auto-PU systems in terms of F-measure, precision and recall. Table 11 has the same structure as Table 5. In each cell comparing the average ranks of a pair of Auto-PU systems, the best (lower) rank is shown in boldface, and if the result is statistically significant (i.e. if the p-value is smaller than the adjusted α) the p-value is also shown in boldface.

The results shown in Table 11 are somewhat mixed for F-measure, with GA-2 performing best overall for δ = 20% and 60%, but worst for δ = 40%, where EBO-2 performs best. Overall, conclusions regarding the best of the three systems are hard to draw from these results, as the results were somewhat mixed across all three measures and all three δ values. The only statistically significant result in Table 11 is that EBO-2 outperformed GA-2 regarding precision for δ = 40%.

Figure 4 shows how the F-measure values achieved by the three Auto-PU systems and the two baseline PU learning

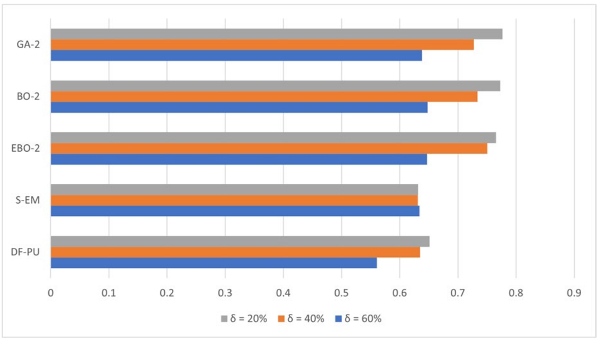


Fig. 4 Average F-measure results comparison for three versions of Auto-PU utilising the base search space and the PU learning baselines across the three values of δ

Table 12 Linear (Pearson's) correlation coefficient value between the F-measure and the percentage of positive examples in the original dataset (before hiding some positive examples in the unlabelled set) for each combination of an Auto-PU system (with extended search space) or PU method and a δ value

Method	$\delta = 20\%$	$\delta = 40\%$	$\delta = 60\%$
EBO-2	0.363	0.361	0.447
BO-2	0.339	0.348	0.225
GA-2	0.340	0.357	0.580
DF-PU	0.381	0.504	0.445
S-EM	0.690	0.631	0.686

methods (DF-PU and S-EM) vary across the three δ values. The results of the baseline methods will be discussed later. As shown in this figure, the F-measure values of GA-2, BO-2 and EBO-2 all decrease with an increase in the δ value, which is consistent with the results obtained with the base search space (Fig. 3). As mentioned earlier, this is not surprising, since δ represents the percentage of positive instances hidden in the unlabelled set; and intuitively, the larger this percentage, the more difficult the PU learning problem is. Again (like in Fig. 3), the decrease in F-measure values was particularly substantial when δ increased from 40% to 60%, for all the three Auto-PU systems.

Table 12 shows the values of Pearson's linear correlation coefficient between the F-measure values of each Auto-PU system or baseline PU learning method and the percentages of positive examples in the original dataset, for each δ value. Table 12 shows the same trends of positive correlations as Table 6– as the percentage of positive instances increases, the F-measure trends upwards. This holds true for all versions of Auto-PU and for all δ values, further implying that PU methods generally perform better when the percentage of positive instances is higher, as expected. The strongest correlation in Table 12 is for GA-2 for δ = 60%. Interestingly, BO-2 shows a weak positive correlation for this δ value. Overall, BO-2 shows a lower positive correlation than the other two methods, albeit by a very small margin for δ = 20% and δ = 40%.

6.2.2 Comparing the Auto-PU Systems against two baseline PU methods

Table 13 shows the statistical significance results of comparing each of the three Auto-PU systems with the extended search space against each of the two baseline PU methods. Table 13 has the same structure as the previously described Table 8. Table 13 shows that the Auto-PU systems significantly outperform the baseline methods in most cases for

Table 13 Results of Wilcoxon signed-rank tests with Holm correction for multiple hypothesis when comparing GA-2, BO-2 and EBO-2 against the two baseline PU methods regarding F-measure, Precision and Recall, for the three δ values

δ(%)	Compared methods	F-measure			Precision			Recall		
		Ranking	P-value	Adj. α	Ranking	P-value	Adj. α	Ranking	P-value	Adj. α
20%	GA-2 vs DF-PU	**1.1** vs 1.9	**3E-05**	0.025	**1.05** vs 1.95	**1E-04**	0.025	**1.45** vs 1.55	0.812	0.05
	GA-2 vs S-EM	**1.2** vs 1.8	**0.002**	0.05	**1.12** vs 1.88	**0.001**	0.05	1.8 vs **1.2**	**0.009**	0.025
	BO-2 vs DF-PU	**1.1** vs 1.9	**4E-05**	0.025	**1.05** vs 1.95	**2E-04**	0.025	**1.45** vs 1.55	0.784	0.05
	BO-2 vs S-EM	**1.25** vs 1.75	**0.003**	0.05	**1.08** vs 1.92	**0.001**	0.05	1.85 vs **1.15**	**0.017**	0.025
	EBO-2 vs DF-PU	1.2 vs 1.8	**2E-04**	0.025	**1.1** vs 1.9	**5E-05**	0.025	1.52 vs **1.48**	0.968	0.05
	EBO-2 vs S-EM	**1.3** vs 1.7	**0.001**	0.05	**1.05** vs 1.95	**8E-05**	0.05	1.8 vs **1.2**	**0.012**	0.025
40%	GA-2 vs DF-PU	**1.3** vs 1.7	**0.027**	0.05	**1.12** vs 1.88	**0.003**	0.025	1.58 vs **1.42**	0.586	0.05
	GA-2 vs S-EM	**1.2** vs 1.8	**0.011**	0.025	**1.18** vs 1.82	**0.005**	0.05	1.85 vs **1.15**	**0.001**	0.025
	BO-2 vs DF-PU	**1.25** vs 1.75	**0.007**	0.025	**1.12** vs 1.88	**0.001**	0.025	1.55 vs **1.45**	0.701	0.05
	BO-2 vs S-EM	**1.25** vs 1.75	**0.008**	0.05	**1.18** vs 1.82	**0.003**	0.05	1.8 vs **1.2**	**0.007**	0.025
	EBO-2 vs DF-PU	**1.2** vs 1.8	**0.001**	0.025	**1.15** vs 1.85	**2E-04**	0.025	1.52 vs **1.48**	0.717	0.05
	EBO-2 vs S-EM	**1.2** vs 1.8	**0.001**	0.05	**1.15** vs 1.85	**5E-04**	0.05	1.75 vs **1.25**	**0.024**	0.025
60%	GA-2 vs DF-PU	**1.3** vs 1.7	0.048	0.025	**1.2** vs 1.8	**0.015**	0.05	1.52 vs **1.48**	0.658	0.05
	GA-2 vs S-EM	**1.4** vs 1.6	0.546	0.05	**1.12** vs 1.88	**0.006**	0.025	1.85 vs **1.15**	**8E-05**	0.025
	BO-2 vs DF-PU	**1.35** vs 1.65	0.070	0.025	**1.15** vs 1.85	**0.005**	0.05	1.5 vs 1.5	0.571	0.05
	BO-2 vs S-EM	1.5 vs 1.5	0.571	0.05	**1.12** vs 1.88	**3E-04**	0.025	1.9 vs **1.1**	**1E-05**	0.025
	EBO-2 vs DF-PU	**1.3** vs 1.7	**0.027**	0.025	**1.4** vs 1.6	0.143	0.05	**1.3** vs 1.7	0.294	0.05
	EBO-2 vs S-EM	1.5 vs 1.5	0.841	0.05	**1.28** vs 1.72	**0.016**	0.025	1.8 vs **1.2**	**0.001**	0.025

Table 14 Average runtime per dataset (in minutes), on a 48-core GPU system, for Auto-PU systems using the extended search space and the two baseline methods.

Method	Avg. Runtime (min)
GA-2	223.2
EBO-2	20.2
BO-2	9.8
DF-PU	4.9
S-EM	1.5

Table 15 Average rank (based on F-meausre values) for each method and p-value for the Friedman test comparing all 8 methods, for each δ value

Method	δ=20%	δ=40%	δ=60%
EBO-1	3.750	3.625	3.950
GA-1	3.675	4.425	3.350
BO-1	3.750	3.600	4.475
EBO-2	4.725	3.700	4.825
GA-2	3.575	4.275	4.200
BO-2	3.500	4.100	4.350
DF-PU	6.825	6.075	6.000
S-EM	6.200	6.200	4.850
Friedman p-value	<0.0001	0.0005	0.0496

F-measure and Precision; whilst the baseline methods significantly outperform the Auto-PU systems in half of the cases for Recall.

As discussed earlier, computational expense is also a factor to consider when utilising Auto-ML systems. The average runtimes of the systems required to complete a nested cross-validation per dataset, running on a 48-core GPU, are summarised in Table 14. BO-2 ran approximately 23 times faster than GA-2 and just over twice as fast as EBO-2. Meanwhile, EBO-2 was about 11 times faster than GA-2.

6.3 Statistical comparison of all methods across both types of search space

This section presents the results of Friedman tests for comparing the F-measure results of all eight methods– i.e., six Auto-PU systems (GA/BO/EBO-based systems for the base and extended search spaces) and the two baseline methods (DF-PU and S-EM)– for each of the three δ values (20%, 40%, 60%). This statistical analysis is intended to complement the statistical analyses conducted separately for the base and extended search space, reported in Sections 6.1 and 6.2.

Table 15 shows the average ranks for all methods, for each δ value. The Friedman test revealed significant differences in performance among methods for each δ value: $p = 0.000001$ for $\delta = 20\%$, $p = 0.000474$ for $\delta = 40\%$, and $p = 0.049643$ for $\delta = 60\%$.

Since the Friedman test yielded significant results for each δ value, we proceeded with post-hoc pairwise method comparisons using the Wilcoxon signed-rank test with Holm correction (as discussed in Section 5.3). For each δ value, all 28 pairs of methods are compared. Consequently, the smallest adjusted significance threshold used for Holm correction was $\alpha = \frac{0.05}{28} = 0.0018$. To conserve space, for each δ value, we show in a table only the pairs of methods whose difference of results was deemed statistically significant, after the Holm correction.

Table 16 Statistically significant pairwise method comparisons for δ =20%, based on Wilcoxon-signed-rank test with Holm correction

Method A	Method B	P-value	Adjusted α
EBO-1	DF-PU	0.0001	0.0020
EBO-1	S-EM	0.0017	0.0025
GA-1	DF-PU	<0.0001	0.0018
BO-1	DF-PU	0.0010	0.0022
EBO-2	DF-PU	0.0002	0.0021
EBO-2	S-EM	0.0014	0.0023
GA-2	DF-PU	<0.0001	0.0019
BO-2	DF-PU	<0.0001	0.0019
BO-2	S-EM	0.0017	0.0024

Table 17 Statistically significant pairwise method comparisons for δ =40%, based on Wilcoxon-signed-rank test with Holm correction

Method A	Method B	P-value	Adjusted α
EBO-1	DF-PU	0.0002	0.0018
EBO-1	S-EM	0.0004	0.0019
GA-1	S-EM	0.0020	0.0021
EBO-2	DF-PU	0.0007	0.0019
EBO-2	S-EM	0.0015	0.0020

For $\delta = 20\%$, nine pairwise comparisons were statistically significant after Holm correction, as shown in Table 16. In all these nine cases, one of the Auto-PU systems (a GA, BO or EBO version, in column Method A) significantly outperformed one of the two baseline methods (DF-PU or S-EM, in column Method B). There was no significant difference between the results of any pair of Auto-PU systems.

For $\delta = 40\%$, five method pairs showed significant differences (Table 17). In all these five cases, one of the Auto-PU systems (Method A) significantly outperformed one of the two baseline methods (Method B). Again, there was no significant difference between the results of any pair of Auto-PU systems.

For $\delta = 60\%$, the Friedman test yielded marginal significance ($p = 0.049643$), but none of the pairwise method comparisons were statistically significant after Holm correction– so, there is no table with the Holm test's results for this δ value.

Overall, EBO achieved the best results in Tables 16 and 17, since both EBO-1 and EBO-2 consistently significantly outperformed both baseline methods for both $\delta = 20\%$ and $\delta = 40\%$, unlike GA and BO.

6.4 Analysis of the most frequently selected hyperparameter values

Tables 18 - 20 show the hyperparameter values most frequently selected by the Auto-PU systems to optimise PU learning algorithms. Each table shows the results for one of the three Auto-PU systems, and for each hyperparameter shown in the first column, there are two rows, reporting the results for the Auto-PU system versions with base or extended search space. The next columns show the most selected value of the hyperparameter, its selection frequency, the baseline frequency and the difference between the selection and baseline frequencies. The selection frequency is the percentage of times that hyperparameter value was selected out of all runs of that version of the Auto-PU system, over all 5 iterations of the external cross-validation procedure for all 20 datasets and all three δ values (i.e. the selection % out of 300 cases). The baseline frequency is the expected selection frequency of a hyperparameter value if all values of that hyperparameter were randomly selected for use in a PU learning algorithm. I.e., it is calculated by dividing 1 (one) by the number of candidate values for that hyperparameter. In Tables 18 - 20 there are no results for the Spy-related hyperparameters ("N/A" = not applicable) in the rows for the base search space because the Spy method is used only in the extended search space.

Table 18 Hyperparameter values most frequently selected by GA-Auto-PU

Hyperparameter	Search space	Most selected value	Selection Freq. (%)	Baseline Freq. (%)	Diff. (%)
Phase 1A Iteration Count	base	1	19.67	10.00	9.67
	extended	1	26.67	10.00	16.67
Phase 1A RN Threshold	base	0.3	15.00	10.00	5.00
	extended	0.3	20.33	10.00	10.33
Phase 1A Classifier	base	Gaussian NB	14.00	5.56	8.44
	extended	Random forest	12.67	5.56	7.11
Phase 1B Flag	base	False	52.33	50.00	2.33
	extended	False	66.00	50.00	16.00
Phase 1B RN Threshold	base	0.15	19.00	10.00	9.00
	extended	0.25	15.33	10.00	5.33
Phase 1B Classifier	base	Deep forest	11.00	5.56	5.44
	extended	SVM	10.67	5.56	5.11
Spy rate	base	N/A	N/A	N/A	N/A
	extended	0.1	18.67	14.29	4.38
Spy tolerance	base	N/A	N/A	N/A	N/A
	extended	0.06	18.47	9.09	9.38
Spy flag	base	N/A	N/A	N/A	N/A
	extended	False	73.33	50.00	23.33
Phase 2 Classifier	base	LDA	21.00	5.56	15.44
	extended	Deep forest	10.00	5.56	4.44

Regarding the Phase 1A Classifier, all Auto-PU systems had a clear preference for simple linear classifiers, with Gaussian naïve Bayes (Table 18), Bernoulli naïve Bayes (Table 19), logistic regression (Tables 19 and 20), and linear discriminant analysis (LDA) (Table 20). These classifiers adhere to the assumptions of separability and smoothness (see Section 2.1), key assumptions for the two-step framework. The exception is random forest, the most frequently selected Phase 1A classifier by GA-2 (Table 18). As random forest is an ensemble classifier, it is neither simple nor linear. However, random forest is a powerful classifier, so it is unsurprising that it was frequently selected. There is little cohesion in the Phase 1B classifier, and as such no clear conclusions can be drawn for that hyperparameter.

There are two reoccurring classifiers selected as the Phase 2 classifier for all Auto-PU systems, LDA (Tables 18 and 19) and the deep forest classifier (Tables 18 and 20). LDA is a relatively simple linear classifier, whilst deep forest is a complex classifier, inspired by deep learning and creating an "ensemble of ensembles" [12]. Deep forest is also the base classifier for one of the baseline approaches used in this work, DF-PU.

Regarding the Phase 1B flag, almost all results show it set to true at a rate between 50% and 60%. As such, the use of Phase 1B is highly dependent on the dataset.

Regarding the Spy flag parameter for all Auto-PU systems, these results show it set to false far more frequently than set to true in all tables. This is surprising, given the prevalence of the spy method throughout the PU learning literature. These results indicate that it may not be as effective as the frequency of its use in the literature would suggest. This further justifies the need for Auto-ML systems such as those proposed in this work, instead of simply relying on popular algorithms in the literature.

Regarding the Phase 1A iteration count hyperparameter, this hyperparameter splits the unlabelled set into multiple subsets in order to prevent the Phase 1A classifier being overwhelmed by the unlabelled set, given that it is generally the majority class. Therefore, the higher the degree of class imbalance, the higher the Phase 1A iteration count hyperparameter should be. In order to evaluate this, the Pearson's correlation coefficient was calculated to analyse the correlation between this hyperparameter and the degree

Table 19 Hyperparameter values most frequently selected by BO-Auto-PU

Hyperparameter	Search space	Most selected value	Selection Freq. (%)	Baseline Freq. (%)	Diff. (%)
Phase 1A Iteration Count	base	2	19.00	10.00	9.00
	extended	2	21.00	10.00	11.00
Phase 1A RN Threshold	base	0.05	14.33	10.00	4.33
	extended	0.25	13.00	10.00	3.00
Phase 1A Classifier	base	Bernoulli NB	8.67	5.56	3.11
	extended	Logistic reg.	8.67	5.56	3.11
Phase 1B Flag	base	True	52.67	50.00	2.67
	extended	True	50.67	50.00	0.67
Phase 1B RN Threshold	base	0.2	14.00	10.00	4.00
	extended	0.2	14.67	10.00	4.67
Phase 1B Classifier	base	HGBoost	8.00	5.56	2.44
	extended	Bagging clas.	7.67	5.56	2.11
Spy rate	base	N/A	N/A	N/A	N/A
	extended	0.3	18.00	14.29	3.71
Spy tolerance	base	N/A	N/A	N/A	N/A
	extended	0.08	12.18	9.09	3.09
Spy flag	base	N/A	N/A	N/A	N/A
	extended	False	74.00	50.00	24.00
Phase 2 Classifier	base	LDA	32.67	5.56	27.11
	extended	LDA	51.67	5.56	46.11

Table 20 Hyperparameter values most frequently selected by EBO-Auto-PU

Hyperparameter	Search space	Most selected value	Selection Freq. (%)	Baseline Freq. (%)	Diff. (%)
Phase 1A Iteration Count	base	2	19.67	10.00	9.67
	extended	2	22.33	10.00	12.33
Phase 1A RN Threshold	base	0.4	13.67	10.00	3.67
	extended	0.15	18.33	10.00	8.33
Phase 1A Classifier	base	Logistic reg.	14.00	5.56	8.44
	extended	LDA	10.33	5.56	4.77
Phase 1B Flag	base	True	59.67	50.00	9.67
	extended	True	56.67	50.00	6.67
Phase 1B RN Threshold	base	0.4	18.00	10.00	8.00
	extended	0.2	14.67	10.00	4.67
Phase 1B Classifier	base	SVM	11.67	5.56	6.11
	extended	Logistic reg.	9.00	5.56	3.44
Spy rate	base	N/A	N/A	N/A	N/A
	extended	0.1	24.00	5.56	18.44
Spy tolerance	base	N/A	N/A	N/A	N/A
	extended	0.01	13.88	10.00	3.88
Spy flag	base	N/A	N/A	N/A	N/A
	extended	False	65.67	50.00	15.67
Phase 2 Classifier	base	Random forest	9.33	5.5%	3.77
	extended	Deep forest	10.67	5.56	5.11

of class imbalance, calculated as the percentage of positive instances in the whole dataset. Table 21 shows the results of this analysis, with all methods showing a moderate to strong negative correlation, as defined by [54]. Based on these correlations, it can be argued that when assembling a two-step PU learning algorithm, one should consider the class imbalance present when deciding upon the value of the iteration count hyperparameter to apply.

7 Conclusions

This work has introduced two new Auto-ML systems for PU learning (referred to as Auto-PU systems), namely BO-Auto-PU and EBO-Auto-PU, based on Bayesian Optimisation and a new hybrid Evolutionary Bayesian Optimisation method, respectively. These two new Auto-PU systems were compared against GA-Auto-PU, the only existing Auto-ML system for PU learning, as well as against two baseline PU learning methods, DF-PU (deep forests adapted to PU learning) and S-EM (the Spy method for PU learning). The evaluation of these Auto-PU systems and PU learning methods was based mainly on the F-measure, which is the most popular measure of predictive performance in PU learning [23]; but we reported statistical significance results not only for F-measure, but also for precision and recall, for the sake of completeness. The experiments considered three versions of each of 20 biomedical datasets, varying the parameter δ, the percentage of positive training examples hidden in the unlabelled training set (δ = 20%, 40%, 60%). Each Auto-PU system

Table 21 Pearson's correlation coefficient values for Phase 1A iteration count to class imbalance

Method	$\delta = 20\%$	$\delta = 40\%$	$\delta = 60\%$
GA-1	-0.646	-0.655	-0.689
GA-2	-0.631	-0.687	-0.723
BO-1	-0.677	-0.700	-0.700
BO-2	-0.641	-0.706	-0.736
EBO-1	-0.656	-0.688	-0.696
EBO-2	-0.680	-0.710	-0.687

has two versions, with the base and extended search spaces, where only the latter includes the Spy method.

The results for the base search space (Section 6.1) have shown that, overall, EBO-1 performed slightly best for δ =20% and δ=40%, whilst GA-1 performed best for δ=60%. In general, however, for all results with the base search space, across the three δ values, the differences in F-measure, precision and recall values among the three systems are not statistically significant. The comparisons between the Auto-PU systems and the baseline methods have shown all three Auto-PU systems largely outperforming the baselines with statistical significance in most cases regarding F-measure and precision.

The results for the extended search space (Section 6.2) show that, among the three Auto-PU systems, GA-2 has achieved overall the best ranks for δ=20% and 60%, whilst EBO-2 has achieved the best ranks for δ=40%. In nearly all cases (combinations of performance measures and δ values), the differences of results among GA-2, BO-2 and EBO-2 was not statistically significant. The exception is that EBO-2 outperformed GA-2 regarding precision with statistical significance for δ = 40%. All three Auto-PU systems often outperformed the two baseline methods with statistical significance for F-measure and precision.

In addition, when comparing all methods (for both search spaces) as a whole, although there were no significant difference between the results of any pair of Auto-PU systems, EBO was the only type of Auto-PU system which consistently outperformed both baseline methods for both search spaces for both $\delta = 20\%$ and $\delta = 40\%$.

Furthermore, for the three Auto-PU systems, there was a moderate degree of correlation between the percentage of positive instances in the dataset and the F-measure values achieved by those systems, whilst there was a stronger correlation for S-EM. DF-PU exhibited a similar correlation to the Auto-PU systems.

Regarding runtime, BO-Auto-PU is about 27 times faster than GA-Auto-PU. EBO-Auto-PU sits between the two other Auto-PU systems at approximately 10 times faster than GA-Auto-PU and almost three times as slow as BO-Auto-PU.

Considering the trade-off between predictive accuracy and computational efficiency, arguably EBO-Auto-PU is the best performing system overall, achieving good predictive performance that is often a statistically significant improvement over the baseline PU learning methods, whilst maintaining a good computational efficiency.

Future work could involve exploring other search spaces, such as a search space incorporating another PU learning framework such as biased learning [1]. In addition, note that, although each Auto-PU system is optimising the hyperparameters of a PU learning algorithm (in the two-step framework), the hyperparameters of each Auto-PU system itself are currently not optimised. In future work the Auto-PU systems' hyperparameters could perhaps be optimised too.

Author Contributions Jack D. Saunders: Conceptualisation, writing– original draft preparation, program code implementation and experimentation. Alex A. Freitas: Conceptualisation, writing– review and editing. All the authors have read and agreed to the published version of the manuscript.

Data Availability Program code for the systems described in this work are available at the GitHub repository: https://github.com/jds39/GA-Auto-PU - The datasets used in this work are available at the GitHub repository: https://github.com/jds39/Unlabelled-Datasets.

Declarations

Competing Interests The authors declare that they have no competing interests.

Ethical and informed consent for data used This article does not involve any studies with human participants or animals performed by any of the authors.

References

1. Bekker J, Davis J (2020) Learning from positive and unlabeled data: A survey. Mach Learn 109(4):719–760
2. Jaskie K, Spanias A (2019) Positive and unlabeled learning algorithms and applications: A survey. In: 2019 10th International conference on information, intelligence, systems and applications (IISA), pp 1–8. IEEE
3. Zhang Y, Li L, Zhou J, et al (2017) Poster: A PU learning based system for potential malicious URL detection. In: Proceedings of the ACM conference on computer and communications security, pp 2599–2601
4. Luo Y, Cheng S, Liu C, et al (2018) PU learning in payload-based web anomaly detection. In: Proceedings of the third international conference on security of smart cities, industrial control system and communications, pp 1–5
5. Yang P, Li X, Mei K et al (2012) Positive-unlabelled learning for disease gene identification. Bioinformatics 28(20):2640–2647
6. Nikdelfaz O, Jalili S (2018) Disease genes prediction by HMM based PU-learning using gene expression profiles. J Biomed Inf 81:102–111
7. Vasighizaker A, Jalili S (2018) C-PUGP: A cluster-based positive unlabeled learning method for disease gene prediction and prioritization. Comput Biol Chem 76:23–31

8. Liu B, Lee WS, Yu PS, Li X (2002) Partially supervised classification of text documents. ICML, vol 2. NSW, Sydney, pp 387–394
9. Ke T, Yang B, Zhen L et al (2012) Building high-performance classifiers using positive and unlabelled examples for text. In: International Symposium on Neural Networks, pp 187–195
10. Liu L, Peng T (2014) Clustering-based method for positive and unlabelled text categorization enhanced by improved TFIDF. J Inf Sci Eng 30:1463–1481
11. Saunders JD, Freitas AA (2022) GA-Auto-PU: A genetic algorithm-based automated machine learning system for positive-unlabeled learning. In: Proceedings of the genetic and evolutionary computation conference companion. GECCO '22, pp 288–291. ACM
12. Zeng X, Zhong Y, Lin W, Zou Q (2020) Predicting disease-associated circular RNAs using deep forests combined with positive-unlabeled learning methods. Briefings in Bioinformatics 21(4):1425–1436
13. Li XL, Zhang L, Liu B, Ng SK (2010) Distributional similarity vs. PU learning for entity set expansion. In: Proceedings of the ACL 2010 conference short papers, pp 359–364
14. Xia R, Hu X, Lu J, et al (2013) Instance selection and instance weighting for cross-domain sentiment classification via PU learning. In: Twenty-third international joint conference on artificial intelligence, pp 2176–2182
15. Ke T, Jing L, Lv H et al (2018) Global and local learning from positive and unlabeled examples. Appl Intell 48:2373–2392
16. Zhang J, Wang Z, Meng J et al (2018) Boosting positive and unlabeled learning for anomaly detection with multi-features. IEEE Trans Multimed 21(5):1332–1344
17. Schrunner S, Geiger B.C, Zernig A, Kern R (2020) A generative semi-supervised classifier for datasets with unknown classes. In: Proceedings of the 35th annual ACM symposium on applied computing, pp 1066–1074
18. Liu B, Liu Z, Xiao Y (2021) A new dictionary-based positive and unlabeled learning method. Applied Intelligence, pp 1–15
19. He Y, Li X, Zhang M, et al (2023) A novel observation points-based positive-unlabeled learning algorithm. Chinese Association for Artificial Intelligence Transactions on Intelligence Technology, pp 1–19
20. Zhou Z, Feng J (2019) Deep forest. National Sci Rev 6(1):74–86
21. Elkan C, Noto K (2008) Learning classifiers from only positive and unlabeled data. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 213–220
22. Japkowicz N, Shah M (2011) Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press
23. Saunders JD, Freitas AA (2022) Evaluating the predictive performance of positive-unlabelled classifiers: a brief critical review and practical recommendations for improvement. ACM SIGKDD Explorations Newsletter 24(2):5–11
24. Yao Q, Wang M, Chen Y, Dai W, Li Y, Tu W, Yang Q, Yu Y (2018) Taking human out of learning applications: A survey on automated machine learning. arXiv:1810.13306
25. He X, Zhao K, Chu X (2021) AutoML: A survey of the state-of-the-art. Knowl-Based Syst 212:106622
26. Zöller M, Huber M (2021) Benchmark and survey of automated machine learning frameworks. J Artif Intell Res 70:409–472
27. Eiben A, Smith J (2003) Introduction to Evolutionary Computing, vol 53. Springer
28. Olson R, Bartley N, Urbanowicz R, Moore J (2016) Evaluation of a tree-based pipeline optimization tool for automating data science. In: Proceedings of the genetic and evolutionary computation conference vol 2016, pp 485–492
29. Sá A, Pinto W.G, Oliveira L, Pappa G (2017) RECIPE: a grammar-based framework for automatically evolving classification pipelines. In: European conference on genetic programming, pp 246–261. Springer
30. Saunders JD, Freitas AA (2022) Evaluating a new genetic algorithm for automated machine learning in positive-unlabelled learning. In: Artificial evolution - 15th international conference. Lecture Notes in Computer Science, vol 14091, pp 42–57. Springer
31. Frazier P (2018) A tutorial on bayesian optimization. arXiv:1807.02811
32. Močkus J (1975) On bayesian methods for seeking the extremum. In: Optimization techniques IFIP technical conference, pp 400–404
33. De Ath G, Everson R, Rahat A, Fieldsend J (2021) Greed is good: Exploration and exploitation trade-offs in bayesian optimization. ACM Trans Evol Learn Optim 1(1):1–22
34. Jin Y (2011) Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm Evol Comput 1(2):61–70
35. Asuncion A, Newman D (2007) UCI machine learning repository
36. Marcus D, Fotenos A, Csernansky J, Morris J, Buckner R (2010) Open access series of imaging studies: longitudinal mri data in nondemented and demented older adults. Journal of Cognitive Neuroscience 22(12):2677–2684
37. Pereira B, Chin S, Rueda O, Vollan H, Provenzano E, Bardwell H, Pugh M, Jones L, Russell R, Sammut S et al (2016) The somatic mutation profiles of 2,433 breast cancers refine their genomic and transcriptomic landscapes. Nat Commun 7(1):1–16
38. Fleming T, Harrington D (1991) Counting Processes and Survival Analysis. Wiley
39. Islam M, Ferdousi R, Rahman S, Bushra H (2020) Likelihood prediction of diabetes at early stage using data mining techniques. In: Computer Vision and Machine Intelligence in Medical Image Analysis, pp 113–125. Springer,
40. Chicco D, Jurman G (2020) Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. BMC Medical Informatics and Decision Making 20(1):1–16
41. Hlavnička J, Čmejla R, Tykalová T, Šonka K, Růžička E, Rusz J (2017) Automated analysis of connected speech reveals early biomarkers of parkinson's disease in patients with rapid eye movement sleep behaviour disorder. Scientific Reports 7(1):1–10
42. Emon M, Keya M, Meghla T, Rahman M, Al Mamun M, Kaiser M (2020) Performance analysis of machine learning approaches in stroke prediction. In: 2020 4th International conference on electronics, communication and aerospace technology (ICECA), pp 1464–1469. IEEE
43. He J, Zhang Y, Li X, Wang Y (2010) Naive bayes classifier for positive unlabeled learning with uncertainty. In: Proceedings of the 2010 SIAM international conference on data mining, pp 361–372. SIAM
44. Basile T, Mauro N, Esposito F, Ferilli S, Vergari A (2017) Density estimators for positive-unlabeled learning. In: International workshop on new frontiers in mining complex patterns, pp 49–64. Springer
45. Shinkins B, Nicholson BD, Primrose J et al (2017) The diagnostic accuracy of a single CEA blood test in detecting colorectal cancer recurrence: Results from the FACS trial. Public Library of Science One 12(3):1–11
46. Beach T, Adler C (2018) Importance of low diagnostic accuracy for early parkinson's disease. Movement Disorders 33(10):1551–1554
47. De Bruyn G, Graviss E (2001) A systematic review of the diagnostic accuracy of physical examination for the detection of cirrhosis. BMC Medical Informatics and Decision Making 1(1):1–11
48. Palmedo H, Bucerius J, Joe A et al (2006) Integrated PET/CT in differentiated thyroid cancer: Diagnostic accuracy and impact on patient management. Journal of Nuclear Medicine 47(4):616–624

49. Nerad E, Lahaye MJ, Maas M et al (2016) Diagnostic accuracy of CT for local staging of colon cancer: a systematic review and meta-analysis. American Journal of Roentgenology 207(5):984–995
50. Zhang Y, Ren H (2017) Meta-analysis of diagnostic accuracy of magnetic resonance imaging and mammography for breast cancer. Journal of Cancer Research and Therapeutics 13(5):862–868
51. Jaskie K, Spanias A (2022) Positive unlabeled learning. Synthesis Lectures Artif Intell Mach Learn 16(1):2–152
52. Wilcoxon F, Katti S, Wilcox R (1963) Critical Values and Probability Levels for the Wilcoxon Rank Sum Test and the Wilcoxon Signed Rank Test vol 1. American Cyanamid
53. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
54. Schober P, Boer C, Schwarte LA (2018) Correlation coefficients: Appropriate use and interpretation. Anesthesia & Analgesia 126(5):1763–1768