Journal of Machine Learning Research 24 (2023) 1-52 Submitted 11/20; Revised 7/23; Published 7/23

Comprehensive Algorithm Portfolio Evaluation using Item Response Theory

Sevvandi Kandanaarachchi sevvandi.kandanaarachchi@data61.csiro.au
CSIRO's Data61
Research Way, Clayton
VIC 3168, Australia

Kate Smith-Miles smith-miles@unimelb.edu.au
School of Mathematics and Statistics
University of Melbourne
Parkville, VIC 3010, Australia

Editor: Marc Schoenauer

Abstract

Item Response Theory (IRT) has been proposed within the field of Educational Psychometrics to assess student ability as well as test question difficulty and discrimination power. More recently, IRT has been applied to evaluate machine learning algorithm performance on a single classification dataset, where the student is now an algorithm, and the test question is an observation to be classified by the algorithm. In this paper we present a modified IRT-based framework for evaluating a portfolio of algorithms across a repository of datasets, while simultaneously eliciting a richer suite of characteristics - such as algorithm consistency and anomalousness - that describe important aspects of algorithm performance. These characteristics arise from a novel inversion and reinterpretation of the traditional IRT model without requiring additional dataset feature computations. We test this framework on algorithm portfolios for a wide range of applications, demonstrating the broad applicability of this method as an insightful algorithm evaluation tool. Furthermore, the explainable nature of IRT parameters yield an increased understanding of algorithm portfolios.

Keywords: Item Response Theory, algorithm evaluation, algorithm portfolios, classification, machine learning, algorithm selection, instance space analysis, explainable algorithm evaluation.

1. Introduction

Evaluating a diverse set algorithms across a comprehensive set of test problems contributes to an increased understanding of the interplay between test problem characteristics, algorithm mechanisms and algorithm performance. Such an evaluation helps determine an algorithm's strengths and weaknesses, and provides a broad overview of the collective capabilities of an algorithm portfolio. The drawback of many studies that evaluate only a small number of algorithms on a limited set of test problems is that they fail to reveal where any algorithm belongs within a state-of-the-art algorithm portfolio's capabilities, or where the unique strengths and weaknesses of algorithms lie considering a diverse range of test problem difficulties and challenges. After several decades of calls for a more "empirical science" of algorithm testing (Hooker, 1994, 1995), research communities in many fields are now pulling together the components needed for rigorous evaluation of algorithms - open source algorithms and shared test problem repositories - that provide the foundation for new methodologies for empirical evaluations (McGeoch, 1996; Hall and Posner, 2010; Smith-Miles et al., 2014; Casalicchio et al., 2019; Bischl et al., 2016).

In this paper we present a framework that evaluates a portfolio of algorithms based on a novel adaptation of Item Response Theory (IRT). The general premise of IRT is that there is a hidden "quality" or a trait, such as verbal or mathematical ability, that cannot be directly measured (Hambleton and Swaminathan, 2013) but can be inferred from responses to well-designed test questions that are suitably difficult and discriminating. A test instrument such as a questionnaire or an exam containing test items is used to capture participant responses. Using the participant responses to the test items, an IRT model is fitted to estimate the discrimination and difficulty of test items and the ability of participants. In an educational setting, the ability relates to the knowledge of the subject matter tested on the exam; the discrimination of test items inform us which items are better at discriminating between strong and weak students; and the difficulty parameters indicate the difficulty of each test item given the response profile from the participants.

IRT's ability to evaluate performance data and obtain useful insights has made it a natural fit for adaptation to the machine learning domain. Martínez-Plumed et al. (2019) used IRT to evaluate the performance of machine learning algorithms (students in the educational analogy) on a single classification dataset (exam), with the individual observations in a classification dataset (exam questions) used to assess algorithm performance. They train and test many classifiers on a single dataset, and obtain insights about the individual observations and about the portfolio of classifiers on that dataset. As a result they obtain a set of classifier characteristic curves for the dataset. Another IRT based evaluation of algorithm portfolios was carried out by Chen et al. (2019). They proposed a model called β^3-IRT, which extends the Beta IRT model for continuous responses discussed by Yvonnick Noel and Bruno Dauvier (2007). Chen et al. (2019) consider new parametrizations so that the resulting item characteristic curves are not limited to logistic curves and use their model to assess machine learning classifiers. They too evaluate an algorithm portfolio on an individual dataset and draw their conclusions about which algorithm is best for a given observation within a dataset. Both Martínez-Plumed et al. (2019) and Chen et al. (2019) investigate IRT on an individual dataset, which we call a test instance.

These exciting directions have motivated us to expand the use of IRT for understanding the strengths and weaknesses of a portfolio of algorithms when applied to *any* dataset, not just a single dataset. In this case, the test instance is an entire dataset comprising observations, and the 'exam' is comprised of many datasets to evaluate the ability of an algorithm. Extending in this direction is important because the limited amount of diversity contained within a single dataset can shed only a limited amount of light on a portfolio of classifiers, and the classifier characteristic curves heavily depend on the dataset. To obtain a better understanding of the strengths and weaknesses of a portfolio of classifiers, indeed any type of algorithm, we need to evaluate the portfolio on a broader range of datasets from diverse repositories. The excellent foundational work showing how IRT models - with both discrete (Martínez-Plumed et al., 2019) and continuous (Chen et al., 2019) performance metrics - can be used to study performance of machine learning algorithms is ripe for extension to see how the insights that can be generated from an IRT perspective compare with recent advances in algorithm portfolio evaluation and construction.

In recent decades the call for a more empirical approach to algorithm testing (Hooker, 1994) has seen efforts to move beyond a standard statistical analysis to evaluate algorithm portfolios, where strong algorithms have best “on-average" performance across a chosen set of test instances. Machine learning approaches such as meta-learning (“learning to learn”) have been used to learn how algorithm portfolios perform based on characteristics of the test instances (Vilalta et al., 2009), with efforts encompassing a large body of research on topics such as algorithm selection, rankings, recommendation, and ensembles to name a few (Lemke et al., 2015). Fŕechette et al. (2016) use Shapley values – a concept from coalition game theory measuring a component’s marginal contribution to the portfolio – to gain insights into the value of an algorithm in a portfolio.

In a related but orthogonal direction, emphasis in the literature on dataset repository design to facilitate unbiased algorithm evaluation is also a growing research area (Marcia and Bernad´ o Mansilla, 2014; Bischl et al., 2016), motivated by the fact that algorithms are frequently claimed to be superior without testing them on a demonstrably broad range of test instances. Demonstrating that a selected set of test instances or datasets is unbiased and sufficiently diverse is one of the major contributions of the Instance Space Analysis methodology (Smith-Miles and Tan, 2012; Smith-Miles et al., 2014; Smith-Miles and Bowly, 2015; Muñoz et al., 2018), developed by Smith-Miles and co-authors by extending Rice’s algorithm selection framework (Rice et al., 1976). A $2D$ instance space is constructed by projecting all test instances into the instance space in a manner that maximize visual interpretation of the relationships between instance features and algorithm performance. The mathematical boundary defining the instance space can be determined, and the diversity of the test instances within the instance space can be scrutinized. Furthermore, the instance space analysis methodology can be used to answer the question posed by the Algorithm Selection Problem (Rice et al., 1976), “Which algorithm is best suited for my problem?”. This aspect is missed by the standard statistical analysis, which focuses on average performances, and leaves hidden the unique strengths and weaknesses of algorithms and relationships to test instance characteristics.

Our main contribution in this paper is proposing a novel framework for evaluating algorithm portfolios across diverse suites of test instances based on IRT concepts. We call this framework AIRT – Algorithmic IRT. The word *airt* is an old Scottish word which means “to guide”. By re-mapping the educational analogies of students, exams and test questions in a manner that is essentially flipped from the original approach on a single dataset of Martínez-Plumed et al. (2019), we propose an inverted model that yields a richer set of evaluation metrics for algorithm portfolios. Adapting continuous IRT models, we introduce measures for quantifying algorithm *consistency*, an algorithm’s *difficulty limit* in terms of the instances it can handle, and the degree of *anomalousness* of an algorithm’s behavior compared to others in the portfolio. We also explore the problem space and find regions of good and bad performance, which are effectively algorithm strengths and weaknesses. These other measures are not computed by standard statistical methodology used for ranking algorithms, nor are they available from the standard IRT mapping (Martínez-Plumed et al., 2019; Chen et al., 2019). For example, the algorithm with the best overall performance on a suite of test problems may not be stable or consistent in the sense that a small change in a test instance may result in large changes in performance. Or there may be an anomalous algorithm that performs well on test instances for which other algorithms perform poorly, and such insights may be lost in standard ‘on-average’ statistical analysis. Indeed, it is AIRT’s focus on revealing insights into algorithm strengths and weaknesses, based on new methods for visual exploratory data analysis from empirical performance data results, that adds significant value beyond standard statistical analysis or algorithm selection studies.

It is worthwhile noting that methodologies in social sciences focus on explanations as opposed to accurate predictions (Shmueli, 2010). As such, quantitative models in social sciences only have a handful of parameters which have meaningful interpretations. Explanations are often linked with causality. Lewis (1986) states "Here is my main thesis: *to explain an event is to provide some information about its causal history."* Miller (2019) presents an argument for linkages with social sciences stating that "the field of explainable artificial intelligence can build on existing research, and reviews relevant papers from philosophy, cognitive psychology/science, and social psychology, which study these topics." Indeed, AIRT is such a linkage. In educational psychometrics IRT is used to explain the student performance in terms of student ability and test item discrimination and difficulty. For example, difficult test items generally yield lower scores than easy test items. Similarly, students with high ability obtain higher scores compared to students with low ability. Thus, IRT model parameters are used to explain the student and test item characteristics and have causal interpretations. These explainable interpretations get translated to the algorithm evaluation setting as follows: problems with high difficulty generally result in low performance values. Algorithms with high difficulty limits can handle harder problems. Algorithms that are consistent give similar results irrespective of the problem difficulty. Anomalous algorithms behave in an unusual fashion by giving better results to harder problems compared to easier problems. We realise these statements are simple and obvious. But that is an attribute of an explanation; Oxford English Dictionary (June 2016) defines it as *a thing which explains, makes clear, or accounts for something*. Therefore, AIRT metrics come from an explainable model in educational psychometrics and contribute to increasing the explainability of algorithm performance.

Beyond insights and explanations however, AIRT can also be used for algorithm selection to construct a strong portfolio. In this paper we compare the predictive power of the AIRT portfolio to others generated by Shapley values (Fŕechette et al., 2016) and best on average performance. The AIRT portfolio showcases algorithm strengths in different parts of the problem space. In addition to introducing these measures that capture different aspects of algorithm performance and constructing algorithm portfolios, we also assess the goodness of the IRT model by comparing the IRT predicted performance with the actual performance. As a further contribution, we make this work available in the R package `airt` (Kandanaarachchi, 2020). Another point of interest is that, unlike in instance space analysis, we do not need to compute test instance features for AIRT, avoiding the additional computational expense, as well as the somewhat arbitrariness of certain feature choices. AIRT computes a 1-dimensional problem space based on dataset difficulty, which is calculated from the performance results of the algorithm portfolio. Characteristics such as algorithm consistency and anomalousness can be calculated as overall characteristics based only on an algorithm's performance metric, while the region of the problem space for which an algorithm shows superiority can be revealed without the need for features. The fact that similar insights can be obtained from the case studies presented in this paper without the need for feature calculation required by instance space analysis is one of the main advantages of AIRT focused on the broader goal of generating insights into algorithm performance, in addition to constructing strong algorithm portfolios, i.e. addressing both questions of which algorithm should be used for a particular instance, and why?

The remainder of the paper is organized as follows: In Section 2 we provide an introduction to polytomous and continuous IRT models and discuss the contextual differences between traditional applications that use IRT for evaluating educational outcomes and adaptations to evaluate algorithms. We then discuss our alternative adaptation, essentially an inverted model, which creates a rich new set of algorithm evaluation metrics defined by reframing the interpretation of the IRT parameters in Section 3. Using these new metrics, we can visualize the strengths and weaknesses of algorithms in the problem space and construct algorithm portfolios using AIRT. Furthermore, to assess the goodness of the models built within our AIRT framework, we define additional measures based on model predicted performance and actual performance on test instances. AIRT expands on the IRT framework to including such enhancements to enable its application to the broader challenge of understanding algorithm strengths and weaknesses. In Section 5 we illustrate the complete functionality of AIRT – including the algorithm metrics, problem space analysis, strengths and weaknesses of algorithms, algorithm portfolio evaluation and model goodness results – using the detailed case study of OpenML-Weka classification algorithms and test instances available at ASlib repository (Bischl et al., 2016). We refer the reader to Appendix A where further results are summarized on nine more case studies using a variety of ASlib scenarios including from satisfiability (SAT) and constraint satisfaction problem domains. These case studies demonstrate the functionality of AIRT as an exploratory data analysis tool for algorithm portfolio evaluation and how the user can construct a competitive algorithm portfolio using AIRT with the objective of minimizing performance gap. Finally, we discuss future work and present our conclusions in Section 6.

2. IRT: Traditional setting and new mapping

Item Response Theory (IRT) (Lord, 1980; Embretson and Reise, 2013; van der Linden and Hambleton, 2013) refers to a family of latent trait models that is used to explain the relationship between unobservable characteristics such as intelligence or political preference and their observed outcomes such as responses to questionnaires. Attributes such as verbal or mathematical ability, racial prejudice and stress proneness, which cannot be measured directly can be modeled as latent variables. The observed outcomes such as test items and questionnaire responses can be explained using latent trait models. IRT builds a connection between the items of a bigger unit such as a test with the participants' latent traits, thus placing each participant in a latent trait continuum. IRT is commonly used in psychometrics (Cooper and Petrides, 2010) and educational testing (Yen, 1986).

2.1 Dichotomous and polytomous IRT models

We introduce some IRT concepts for dichotomous and polytomous models using the notation of Chalmers (2012) and Rizopoulos (2006). Let $i = 1, \dots N$ represent participants or testees, $j = 1, \dots n$ represent the test items with $N > n$, and let θ denote the latent variable such as intelligence or ability. An example includes a test with n questions, which is administered to a class of N students with the aim of measuring their ability θ to perform certain tasks. The response of the i^{th} participant for the j^{th} item is denoted by x_{ij}. The discrimination parameter for test item j is denoted by α_j and the difficulty parameter by d_j. These two parameters are used to build the 2-Parameter Logistic (2PL) model, while an additional guessing parameter γ_j is incorporated in the 3-Parameter Logistic (3PL) model.

For dichotomous data researchers are interested in modeling the probability of correct response for each item given the ability level θ_i. The 3PL model defines the probability of a correct response for participant i for item j as

$$\Phi\left(x_{ij} = 1 | \theta_i, \alpha_j, d_j, \gamma_j\right) = \gamma_j + \frac{1 - \gamma_j}{1 + \exp\left(-D\alpha_j\left(\theta_i - d_j\right)\right)} \tag{1}$$

where D is the scaling adjustment traditionally set at 1.702. The role of D is to make the logistic curve similar to the cumulative distribution function of the normal distribution (Reckase, 2009). Figure 1 shows the resulting probability for a given item j with fixed α, d and γ disregarding the scaling constant D. The greater the ability θ_i of the participant, the higher the probability of the correct response.

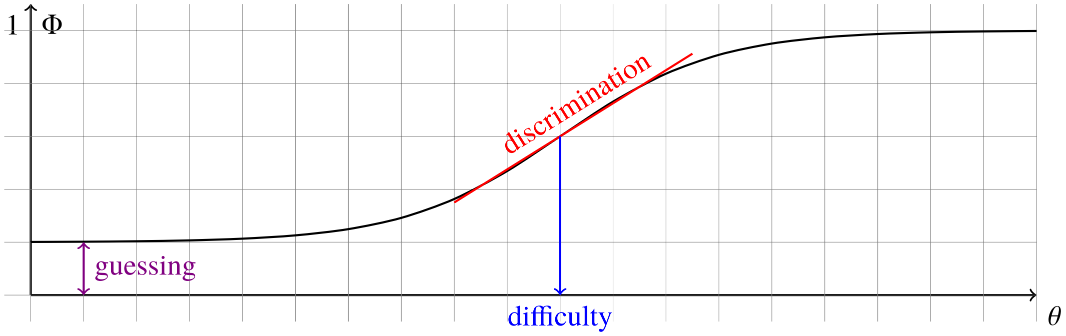


Figure 1: Probability of a correct response for a given item using a 3PL model. Difficulty corresponds to d_j, discrimination to α_j and guessing to γ_j in equation (1).

For polytomous data, we briefly present the multi-response ordinal models described in Samejima (1969). For example, self-esteem surveys have questions such as *I feel that I am a person of worth, at least on an equal plane with others* with responses {*strongly disagree, disagree, neutral, agree, strongly agree*}. In this case the original responses, which are the participants answers, are used to fit the IRT model (Gray-Little et al., 1997). By definition ordinal responses are ordered, i.e., *strongly disagree* < *disagree* < *neutral* < *agree* < *strongly agree*. The responses need to be ordinal because the resulting latent trait continuum is ordered from low ability to high ability. In educational testing an accuracy measure such as marks, derived from the original responses are used to fit the IRT model. For example, for each question in a test, the participants write their answers and marks are derived by the person who grades them. For simplicity, suppose the marks for each question can take the values $\{0, 1, 2, 3, 4, 5\}$. The marks, which is a derived accuracy measure are the responses in this case and is used to fit the IRT model. Similarly, for multiple choice questions with marks taking the values $\{0, 1\}$ a dichotomous IRT model is fitted. Whether a derived accuracy measure or the original responses are used, these are called *responses* in IRT literature. We note that the word *response* is confusing to non-IRT researchers when it refers to grades or other type of measures derived from the original responses. However, as this is the standard term used in IRT literature, we will use the same for easier cross-referencing. If there are C_j unique response categories for item j with $0 < 1 < \cdots < C_j - 1$, difficulty parameters $\boldsymbol{d}_j = \left(d_1, \ldots, d_{C_j - 1}\right)$ and discrimination parameter α_j, the cumulative probabilities are defined as

$$\begin{aligned}
\Phi\left(x_{ij} \geq 0 | \theta_i, \alpha_j, \boldsymbol{d}_j\right) &= 1\,, \\
\Phi\left(x_{ij} \geq 1 | \theta_i, \alpha_j, \boldsymbol{d}_j\right) &= \frac{1}{1 + \exp\left(-D\alpha_j\left(\theta_i - d_1\right)\right)}\,, \\
\Phi\left(x_{ij} \geq 2 | \theta_i, \alpha_j, \boldsymbol{d}_j\right) &= \frac{1}{1 + \exp\left(-D\alpha_j\left(\theta_i - d_2\right)\right)}\,, \\
&\vdots \\
\Phi\left(x_{ij} \geq C_j - 1 | \theta_i, \alpha_j, \boldsymbol{d}_j\right) &= \frac{1}{1 + \exp\left(-D\alpha_j\left(\theta_i - d_{C_j - 1}\right)\right)}\,, \\
\Phi\left(x_{ij} \geq C_j | \theta_i, \alpha_j, \boldsymbol{d}_j\right) &= 0\,,
\end{aligned}$$

where x_{ij} is the response of participant i for question j. This gives the probability of the response $x_{ij} = k$ as

$$\Phi\left(x_{ij} = k | \theta_i, \alpha_j, \boldsymbol{d}_j\right) = \Phi\left(x_{ij} \geq k | \theta_i, \alpha_j, \boldsymbol{d}_j\right) - \Phi\left(x_{ij} \geq (k+1) | \theta_i, \alpha_j, \boldsymbol{d}_j\right)\ .$$

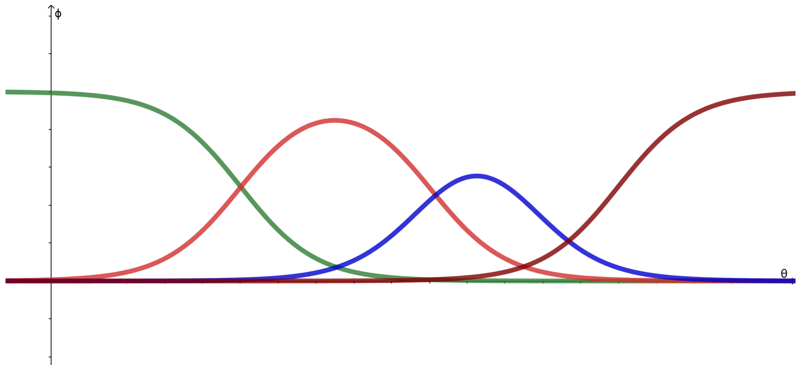


Figure 2: The probability of the response $x_{ij} = k$ for different $k \in \{0, 1, 2, 3\}$ with θ on the horizontal axis and Φ on the vertical axis. The most likely outcome is different depending on the ability levels θ.

Figure 2 shows the probability density functions for different responses $x_{ij} = k$ for $k \in \{0, \ldots, (C_j - 1)\}$. In the educational testing scenario discussed above, each curve denotes the probability that marks are equal to k for $k \in \{0, 1, 2, 3\}$. From Figure 2 we see that the green curve, which gives the probability density function for marks = 0, has high probability when the participant ability θ is low. Similarly, the dark red curve corresponding to marks = 3 has a higher probability for high participant ability. We see that a participant with a lower ability/latent trait is more likely to obtain a response corresponding to a low value of k compared to a participant with a higher ability.

2.2 Continuous IRT models

In addition to the polytomous IRT models, Samejima (1973, 1974) introduced Continuous Response Models (CRM) to extend polytomous models to continuous responses. Wang and Zeng (1998) introduced an expectation-maximization (EM) algorithm for Samejima's continuous item response model. This EM algorithm was further optimized by Shojima (2005) by proposing a non-iterative solution for each EM cycle.

In this section we use the notation used by Wang and Zeng (1998) and Shojima (2005). They consider N examinees with trait variables θ_i where $i \in \{1, \dots, N\}$ and n test instances with parameters $\boldsymbol{\lambda}_j = (\alpha_j, \beta_j, \gamma_j)^T$ for $j \in \{1, \dots, n\}$. The item parameters α_j represents discrimination, β_j difficulty and γ_j a scaling coefficient that defines a scaling transformation from the original rating scale to the θ scale.

Using the normal density type CRM, Wang and Zeng (1998) considered the probability of an examinee with an ability θ obtaining a score of y_j or higher on a given item j as

$$P\left(Y \geq y_j | \theta\right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{v} e^{-\frac{t^2}{2}} dt \,, \tag{2}$$

where

$$v = \alpha_j \left(\theta - \beta_j - \gamma_j \ln \frac{y_j}{k_j - y_j} \right) ,$$

and the continuous score range of y_j is $(0, k_j)$. The continuous score range of $(0, k_j)$ is opened up to $(-\infty, \infty)$ with the reparametrization

$$z_j = \ln \frac{y_j}{k_j - y_j} \,.$$

Using this reparametrization they obtain the probability density function $f\left(z_j|\theta\right)$ by differentiating the cumulative density function obtained using equation (2) as

$$f\left(z_j|\theta\right) = \frac{d}{dz_j}\left(1 - P(Z \geq z_j|\theta)\right) = \frac{\alpha_j \gamma_j}{\sqrt{2\pi}} \exp\left(-\frac{\alpha_j^2}{2}\left(\theta - \beta_j - \gamma_j z_j\right)^2 \right) . \tag{3}$$

Comparing the parameters α_j, β_j and γ_j with those of Section 2.1 we note that the parameter α_j denotes discrimination as in Section 2.1 and the parameter β_j denotes the difficulty level of item j, which was denoted by d_j in Section 2.1. However, the parameter γ_j is quite different to the guessing parameter used in Section 2.1, in that it denotes a scaling factor which we will inspect soon.

For every $z \in \mathbb{R}$, there is an associated probability density function given by $f(z|\theta)$. Figure 3 shows the item response functions obtained for $z \in \{-2, 0, 1\}$ for different items, which have different CRM parameters. Figure 4 shows the heatmap of $f(z|\theta)$ for the same items for continuous z and θ values. The first pane in both Figures show the curves/heatmap for the first item, HaifaCSP-free, with $\alpha = 1.73$, $\beta = 1.16$ and $\gamma = 2.72$. The second item, iZplus-free, has CRM parameters $\alpha = 0.65$, $\beta = 2.6$ and $\gamma = 1.65$. The third item, MZN/Gurobi-free, has CRM parameters $\alpha = 1.14$, $\beta = 1.15$ and $\gamma = 2.49$. We will give more context on these items later. The second item has a higher difficulty level compared to the first and the third we see that $\beta = 2.6$ shifts the curves to the right in Figure 3 and the high density regions have moved to the right in Figure 4. The first item has higher α values making the curves steeper in Figure 3 compared with items 2 and 3. Similarly, the high density regions are narrower and sharper in Figure 4 due to higher discrimination.

Wang and Zeng (1998) estimated the item parameters used in equation (3) using an EM algorithm. Shojima (2005) enhanced the algorithm by proposing a non-iterative step for the expectation cycle,

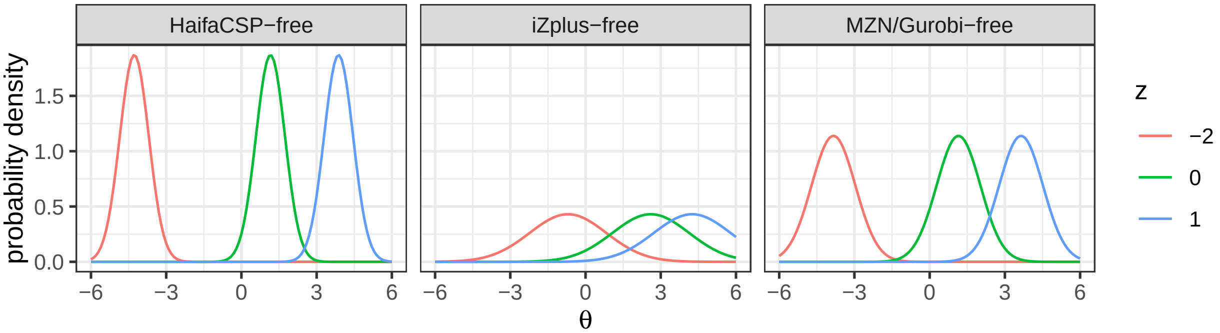


Figure 3: Probability density curves for $z = -2$, $z = 0$ and $z = 1$ for three items with different CRM parameters. The items are from CSP-Minizinc-2016 algorithm portfolio.

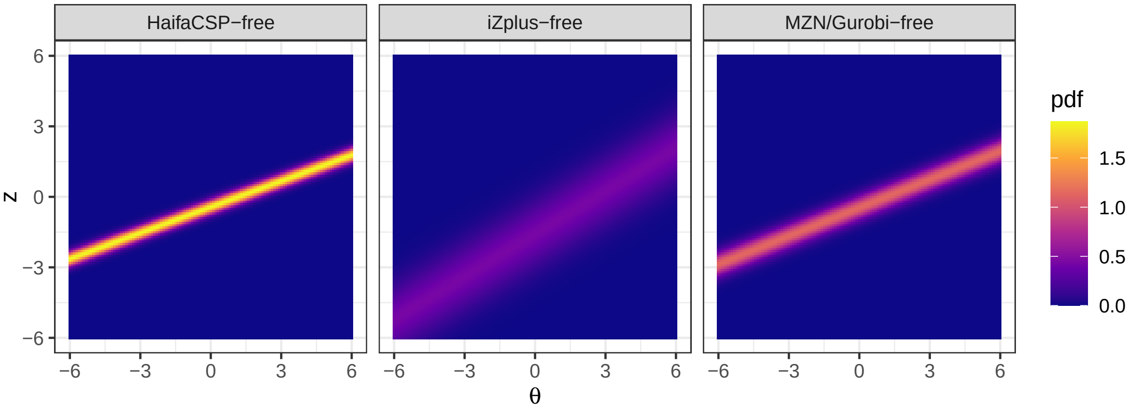


Figure 4: The heatmap of probability density functions for the items in Figure 3

which made the item parameter computation much faster. However, in their estimation Shojima (2005) only considers $\alpha, \gamma > 0$. As such, their algorithm does not accommodate negative discrimination items. This reflects the current practice regarding negative discrimination items in educational and psychometric testing. Negative discrimination items are generally considered as non-value adding and as such revised or removed in traditional educational testing (Hambleton and Swaminathan, 2013). However, in algorithm performance negative discrimination plays an important role and we do not remove such items from the pool.

We accommodate negative discrimination items by modifying the existing algorithm discussed by Shojima (2005). Before discussing these modifications we give a brief overview of their method. First they rescale y_{ij}, such that $x_{ij} = y_{ij}/k_j$ lies in $(0, 1)$ and consider $z_{ij} = \ln x_{ij}/(1 - x_{ij})$. They denote the item response vector of examinee i by z_i. Then they perform a marginal maximum likelihood estimation with the expectation maximization algorithm (MML-EM). Using a normal prior for θ_i, i.e. $\mathcal{N}(\theta_i|\mu, \sigma)$ they obtain an estimate for the posterior distribution of θ_i, given z_i and the current estimates of item parameters as

$$p\left(\theta_i|\boldsymbol{\Lambda}^{(t)},\boldsymbol{z}_i\right) = \mathcal{N}\left(\theta_i|\mu_i^{(t)},\sigma^{(t)2}\right) ,$$

where $\boldsymbol{\Lambda}^{(t)} = \left(\boldsymbol{\lambda}_{\mathbf{1}}^{(t)},\ldots,\boldsymbol{\lambda}_{\boldsymbol{n}}^{(t)}\right)$, $\boldsymbol{\lambda}_{\boldsymbol{j}}^{(t)} = \left(\alpha_j^{(t)},\beta_j^{(t)},\gamma_j^{(t)}\right)^T$ and (t) denotes the iteration. The parameters $\mu_i^{(t)}$ and $\sigma^{(t)}$ are given by

$$\sigma^{(t)2} = \left(\sum_j \alpha_j^{(t)2} + \sigma^{-2}\right)^{-1} ,$$

$$\mu_i^{(t)} = \sigma^{(t)2}\left(\sum_j \alpha_j^{(t)2}\left(\beta_j^{(t)} + \gamma_j^{(t)} z_{ij}\right) + \mu\right) ,$$

where μ and σ denote the initial prior parameters of θ_i. Then they obtain the expectation of the log-likelihood

$$E_{\boldsymbol{\theta}|\Lambda^{(t)},\boldsymbol{Z}}\left[\ln p\left(\boldsymbol{\Lambda}|\boldsymbol{\theta},\boldsymbol{Z}\right)\right] = N\sum_{j=1}^{n}\left(\ln\alpha_j + \ln\gamma_j\right) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{n}\alpha_j^2\left(\left(\beta_j + \gamma_j z_{ij} - \mu_i^{(t)}\right)^2 + \sigma^{(t)2}\right) + \ln p\left(\boldsymbol{\Lambda}\right) + \text{const} \tag{4}$$

where $\boldsymbol{Z}$ is the item response matrix of all examinees for n items and p denotes the probability. They optimize this expectation with flat priors for item parameters and obtain

$$\gamma_j^{(t+1)} = \frac{V\left(\mu_i^{(t)}\right) + \sigma^{(t)2}}{C_j\left(z_{ij},\mu_i^{(t)}\right)} , \tag{5}$$

$$\beta_j^{(t+1)} = M\left(\mu_i^{(t)}\right) - \gamma_j^{(t+1)} M_j\left(z_{ij}\right) , \tag{6}$$

$$\alpha_j^{(t+1)} = \left(\gamma_j^{(t+1)2} V_j(z_{ij}) - V\left(\mu_i^{(t)}\right) - \sigma^{(t)2}\right)^{-1/2} , \tag{7}$$

where M, V and C denote the mean, variance and covariance terms defined by

$$M_j\left(z_{ij}\right) = \frac{\sum_i z_{ij}}{N} ,$$

$$M\left(\mu_i^{(t)}\right) = \frac{\sum_i \mu_i^{(t)}}{N} ,$$

$$V\left(z_{ij}\right) = \frac{\sum_i z_{ij}^2}{N} - M_j\left(z_{ij}\right)^2 ,$$

$$V\left(\mu_i^{(t)}\right) = \frac{\sum_i \mu_i^{(t)2}}{N} - M\left(\mu_i^{(t)}\right)^2 ,$$

$$\text{and} \qquad C_j\left(z_{ij},\, \mu_i^{(t)}\right) = \frac{\sum_i z_{ij}\mu_i^{(t)}}{N} - M_j\left(z_{ij}\right) M\left(\mu_i^{(t)}\right) . \tag{8}$$

For each iteration using $\mu_i^{(t)}$, $\sigma^{(t)}$, and M, V and C quantities listed above, the parameter $\gamma_j^{(t+1)}$ is computed as in equation (5). This value of $\gamma_j^{(t+1)}$ is used to compute $\beta_j^{(t+1)}$ and $\alpha_j^{(t+1)}$ in equations (6) and (7). Using the parameter values $\alpha_j^{(t+1)}$, $\beta_j^{(t+1)}$ and $\gamma_j^{(t+1)}$, the log-likelihood given in equation (4) is computed. This whole process is repeated until the difference in log-likelihoods for successive iterations becomes smaller than a predefined level of convergence. We note that this is a brief overview of this method and refer to Shojima (2005) for more details.

With the current formulation we see that if $\gamma_j^{(t+1)}$ in equation (5) is negative due to a negative covariance term $C_j\left(z_{ij}, \mu_i^{(t)}\right)$ computed as in equation (8), this results in the log-likelihood in equation (4) being incalculable as it requires $\ln \gamma_j$. This forces the MML-EM algorithm to stop, preventing convergence. As a result, this formulation only works when all test instances have $\alpha_j > 0$ and $\gamma_j > 0$ as permitted by the assumption.

However, we see that the probability density function $f(z_j|\theta)$ in equation (3) contains the product $\alpha_j\gamma_j$ and is valid when both α_j and γ_j have the same sign. Similarly, equation (4) can be rewritten with the product $\ln\left(\alpha_j\gamma_j\right)$ instead of the sum of log terms and is valid when both α_j and γ_j have the same sign.

Therefore, if we remove the assumption used by Shojima (2005), that $\alpha_j > 0$ and $\gamma_j > 0$ and update it with $\alpha_j\gamma_j > 0$, we incorporate test items with $\alpha_j, \gamma_j < 0$ as well as test items with $\alpha_j, \gamma_j > 0$. That is, effectively we are adding the assumption $\text{sign}(\alpha_j) = \text{sign}(\gamma_j)$, instead of $\alpha_j > 0$ and $\gamma_j > 0$. More importantly, we are opening the IRT model to negative discrimination items.

With the updated assumption we can rewrite the log-likelihood as

$$E_{\boldsymbol{\theta}|\Lambda^{(t)},\boldsymbol{Z}}\left[\ln p\left(\boldsymbol{\Lambda}|\boldsymbol{\theta},\boldsymbol{Z}\right)\right] = N\sum_{j=1}^{n}\left(\ln|\alpha_j| + \ln|\gamma_j|\right) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{n}\alpha_j^2\left(\left(\beta_j + \gamma_j z_{ij} - \mu_i^{(t)}\right)^2 + \sigma^{(t)2}\right) + \ln p\left(\boldsymbol{\Lambda}\right) + \text{const}, \tag{9}$$

making the log-likelihood tractable for any α_j and γ_j. Then following through the computation we obtain

$$\alpha_j^{(t+1)} = \text{sign}\left(\gamma_j^{(t+1)}\right)\left(\gamma_j^{(t+1)2}V_j(z_{ij}) - V\left(\mu_i^{(t)}\right) - \sigma^{(t)2}\right)^{-1/2}.$$

The parameters $\gamma_j^{(t+1)}$ and $\beta_j^{(t+1)}$ stay the same as given by equations (6) and (7) with the updated assumption. These modifications allow us to fit both negative and positive discrimination items in our continuous IRT model.

The causal interpretation of traditional IRT presumes that the attributes of participant i and test question j give rise to marks x_{ij}. The attributes are the discrimination and difficulty parameters of question j and the ability of the participant i. This is shown in the Directed Acyclic Graph (DAG) in Figure 5. While traditional IRT texts do not include DAGs, more recent work (Kelly et al., 2023) makes these causal interpretations explicit.

2.3 Applications to machine learning and algorithm evaluation

In the traditional IRT setting N participants' responses for n test instances are used to fit an IRT model and obtain the discrimination and difficulty of test instances as well as the ability of the participants. A natural way to use the IRT framework on algorithms and test instances is to consider an algorithm as a participant and test instances as test questions/items. If we formulate our problem this way, then we can obtain the test instance characteristics difficulty and discrimination using the

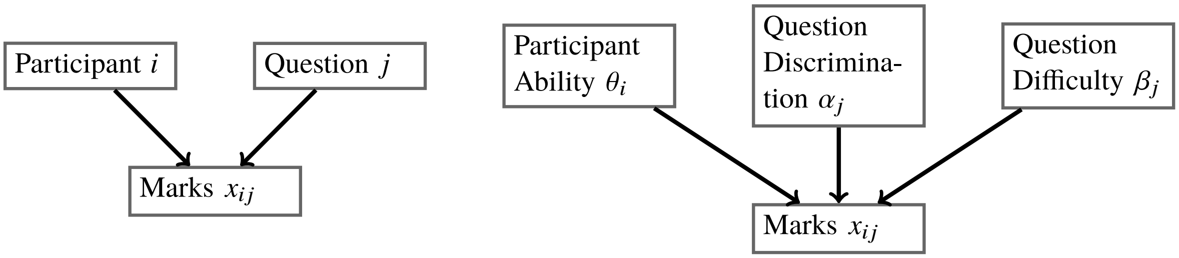
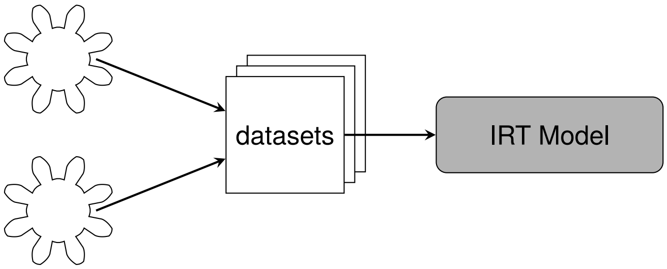

Participant i
Question j
Marks x_{ij}
Participant Ability θ_i
Question Discrimination α_j
Question Difficulty β_j
Marks x_{ij}

Figure 5: Left: A DAG showing participant i and question j giving rise to marks x_{ij}. Right: The DAG composed of participant and question attributes.



Dataset characteristics	Algorithm characteristics
Difficulty Discrimination	Ability

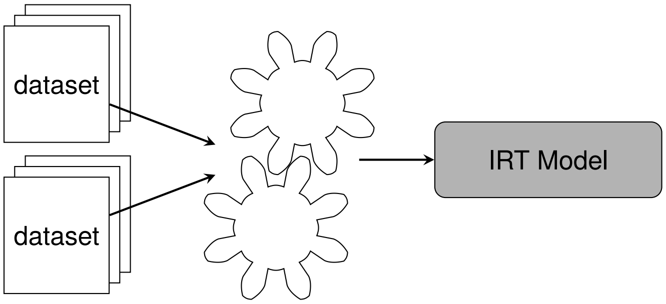
Figure 6: Standard IRT setting extended to algorithms working on datasets. The IRT model provides the dataset characteristics of difficulty and discrimination, and algorithm ability, as outputs.

IRT framework. In addition, IRT will also give us the latent scores or the ability of the algorithms. Martínez-Plumed et al. (2019) and Chen et al. (2019) formulated their problem this way and used the IRT framework to evaluate observations in a dataset and obtain the ability of the classifiers for that dataset.

Instead of using observations of a given dataset as test items, we can also use datasets as test items. Then the parameters fitted by the IRT model would be dataset difficulty and discrimination. This is illustrated in Figure 6. Recent investigations (Kandanaarachchi, 2022) showed the benefits of a flipped approach in constructing an unsupervised anomaly detection ensemble for a single dataset where observations were used as participants and algorithms as test items. In the current paper, we explore this idea further for evaluating algorithms on many datasets, developing a full theory and framework for comprehensive algorithm evaluation.

3. Algorithmic IRT (AIRT)

As a novel adaptation in this paper we now invert the intuitive IRT mapping discussed in the previous paragraph and consider algorithms as items and test instances as participants. This is shown in Figure 7. This inversion results in a loss of intuition momentarily. However, by persisting with this less intuitive mapping we gain an elegant reinterpretation of the theory that enables us to analyze the strengths and weaknesses of algorithms with far more nuanced detail. Firstly, we note that this inversion produces two parameters describing algorithm properties compared to a single parameter in the standard setting. As we will see shortly, we will derive three algorithm characteristics from these two algorithm parameters. Thus, the inversion serves to offer a richer set of metrics with which to evaluate algorithms, compared to the standard approach, which focuses more on dataset/observation evaluation. Table 1 compares the classic IRT approach with the standard and the inverted IRT approaches for algorithm evaluation. With this mapping, we presume that attributes of the problem/dataset i and algorithm j give rise to the performance x_{ij} as shown in the DAG in Figure 8.



Algorithm characteristics	Dataset characteristics
Algo-Char-1 Algo-Char-2	Dataset-Char-1

Figure 7: Inverted IRT setting with datasets acting on algorithms. The IRT model provides two algorithm characteristics in place of difficulty and discrimination, and one dataset characteristic in place of ability as outputs.

Table 1: A comparison between the classic IRT with the standard and inverted IRT approaches for algorithm evaluation.

	Classic IRT	Standard Approach for Algorithm Evaluation	Inverted Approach for Algorithm Evaluation	Inverted Characteristics
Setting	Examinees doing test items	Algorithms working on datasets	Datasets acting on algorithms	
Parameters	Test item difficulty	Dataset difficulty	Difficulty parameter for algorithms	Algorithm difficulty limit
	Test item discrimination	Dataset discrimination	Discrimination parameter for algorithms	Algorithm anomalousness and consistency
	Examinee ability	Algorithm ability	Ability trait of datasets	Dataset difficulty

The inherent meaning of resulting IRT parameters and latent scores is changed when we map algorithms to items and test instances to participants. For example, suppose Figure 9 originates from an educational testing scenario. It shows the heatmap of a test question, the set of trace lines with P1 $<$ P2 $<$ P3 $<$ P4 and a histogram of latent scores. The y-axis in the heatmap labeled z denotes the normalized score and examinee's ability is denoted by θ. Then, as the examinee's ability increases, the probability of getting a better grade for this particular question also increases as seen from the heatmap and the trace lines. For algorithm evaluation, let us also consider the performance levels P1 $<$ P2 $<$ P3 $<$ P4 with higher levels and larger z values indicating better performance. If we consider the standard IRT approach discussed in Table 1, then the heatmap and the trace lines give the performance of a specific dataset and the histogram of latent scores give algorithm abilities.

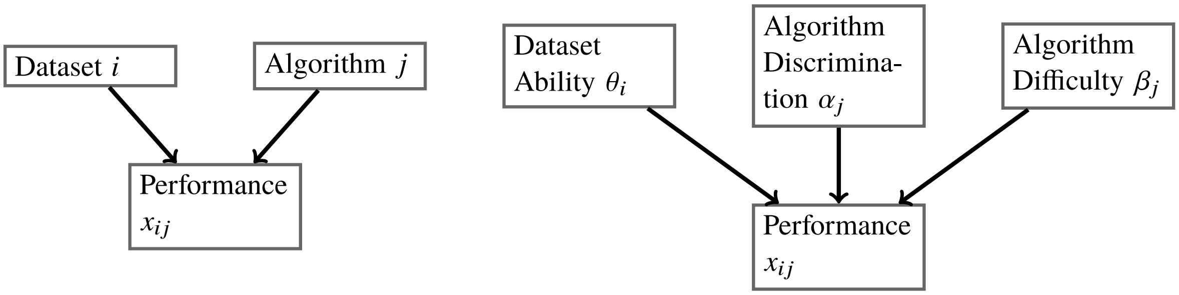

Dataset i
Algorithm j
Performance x_{ij}
Dataset Ability θ_i
Algorithm Discrimination α_j
Algorithm Difficulty β_j
Performance x_{ij}

Figure 8: Mapping IRT to the algorithm evaluation domain, a participant is mapped to a dataset and a question is mapped to an algorithm. Left: The resulting DAG from the this mapping. Right: The DAG composed of dataset and algorithm attributes.

If we consider the inverted IRT approach, the heatmap and the tracelines show the performance of an algorithm and the histogram gives the latent scores of the datasets. What do these latent scores represent? We know that algorithms gives better performance on easy test instances. For example a classification algorithm such as logistic regression will give better classification accuracy on a linearly separable dataset compared to a complex dataset. As such, in the inverted algorithm evaluation setting the latent score θ represents the easiness of the test instance.

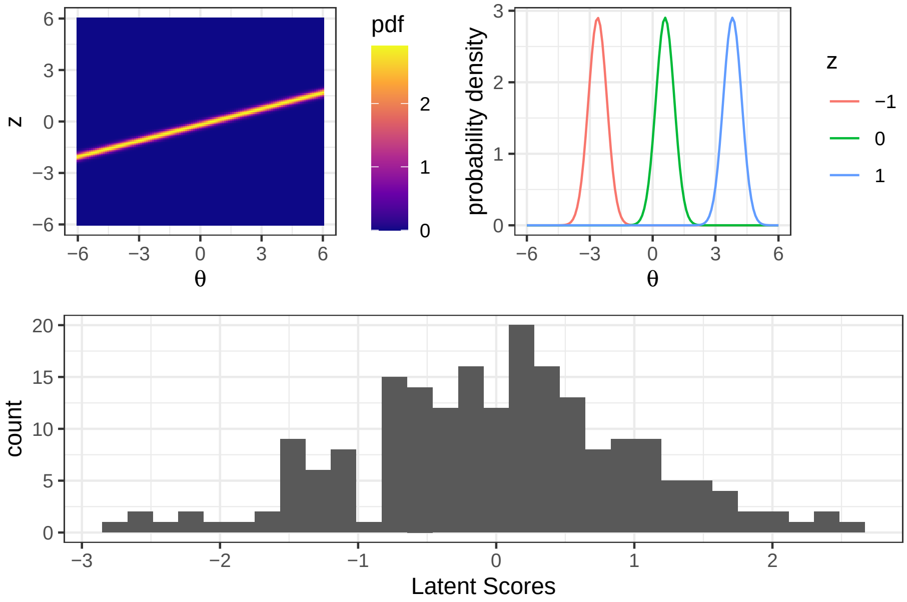


Figure 9: The heatmap of LCG-Glucose-free on the top left and the trace lines for $z \in \{-1, 0, 1\}$ for that item on the top right. The histogram of the latent scores estimated by the model is shown at the bottom. In the inverted IRT algorithm evaluation setting, the latent scores represent test instance easiness.

Furthermore, this inverted setting gives rise to important algorithm characteristics that can now be measured using the IRT parameters, as described in the following sections.

3.1 Framework

Our algorithmic IRT (AIRT) framework consists of three main stages:

1. Stage 1: Fitting an IRT model with inverted mapping
   We input the performance results of n algorithms on N test instances to a continuous or a polytomous IRT model, mapping test instances to participants and algorithms to items. The R package `airt` fits the continuous IRT models described in Section 2.2 using the updated log-likelihood function and assumption. To fit polytomous models `airt` uses the functionality of the existing R package `mirt` (Chalmers, 2012).

2. Stage 2: Calculation of algorithm and dataset metrics
   The second stage consists of reinterpreting the results of the IRT model, due to the inverted mapping and inherent contextual differences, so that a richer set of metrics for algorithm performance and dataset difficulty can be calculated.

3. Stage 3: Compute strengths and weaknesses and construct algorithm portfolios
   Construct latent trait curves to enable algorithm ranking and strengths and weaknesses of algorithm portfolios to be observed across test suites of varying difficulty.

A range of indicators are computed as additional measures that characterize algorithms and assess the goodness of the IRT model, as presented in the following sections. AIRT is applicable to both continuous and polytomous IRT models. Our results on various algorithm portfolios used to validate the approach in Section 5 and Appendix A focus on continuous IRT models, however we note that AIRT can be used to construct polytomous models. We present results for continuous scenarios because they have higher variation and as such are more interesting. We note that the R package `airt` has the functionality to handle polytomous data as well as continuous, and details of the generalization to polytomous data are provided in Supplementary Materials.

We will use CSP-Minizinc-2016 algorithm portfolio from ASlib repository (Bischl et al., 2016) to illustrate algorithm and dataset metrics. For all algorithms in the ASlib repository certain hyperparameters and parameters were used which we do not vary. Any conclusions we draw about algorithm performance are therefore dependent on the actual algorithm implementation they use. Further conclusions about the strengths and weaknesses of any algorithm would need to thoroughly explore the impact of its parameter values.

CSP-Minizinc-2016 contains the results of constrained satisfaction and optimization problems. The original dataset contains the runtimes of each problem instance. As the IRT framework denotes good performance by increasing values we have taken the reciprocal of the runtimes to fit the AIRT model. Figure 10 shows the heatmaps of the probability density functions for all algorithms in the portfolio. The items discussed in Figures 3 and 4 were algorithms taken from this portfolio.

3.2 Dataset metric: Difficulty score

As discussed previously, the latent trait denoted by θ corresponds to dataset easiness and is given by

$$\theta_i = \frac{\sum_j \hat{\alpha}_j^2 \left(\hat{\beta}_j + \hat{\gamma}_j z_{ij} \right)}{\sum_j \hat{\alpha}_j^2} , \tag{10}$$

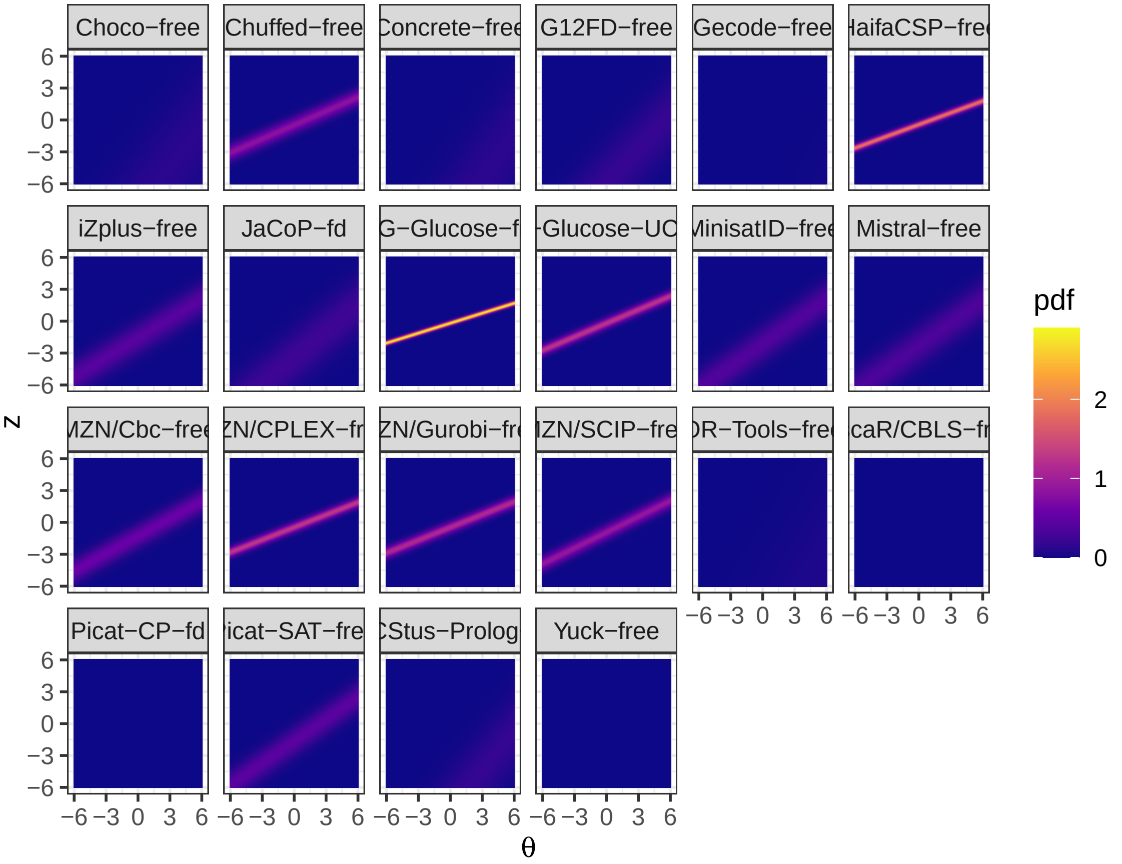


Figure 10: The heatmap of probability density functions for all algorithms in CSP-Minizinc-2016 portfolio.

where $\hat{\alpha}_j$, $\hat{\beta}_j$ and $\hat{\gamma}_j$ are the estimated discrimination, difficulty and scaling parameters for algorithm j, which are obtained by fitting the IRT model. Using θ_i we define dataset difficulty as

$$\delta_i = -\theta_i \, , \tag{11}$$

where δ_i denotes the difficulty of the i^{th} dataset. We see that dataset difficulty is a function of discrimination, difficulty and scaling parameters of algorithms as well as the accuracy scores of the datasets.

Shojima (2005) uses the normal density type CRM with normal priors making the posterior distribution of the trait parameter θ normal. Thus, we can expect dataset difficulty δ to be normally distributed. We refer to datasets/problems as easy if they have low difficulty values. Similarly, we say datasets/problems are difficult if they have high difficulty values. The semi-difficult or semi-easy instances are in the middle of the spectrum.

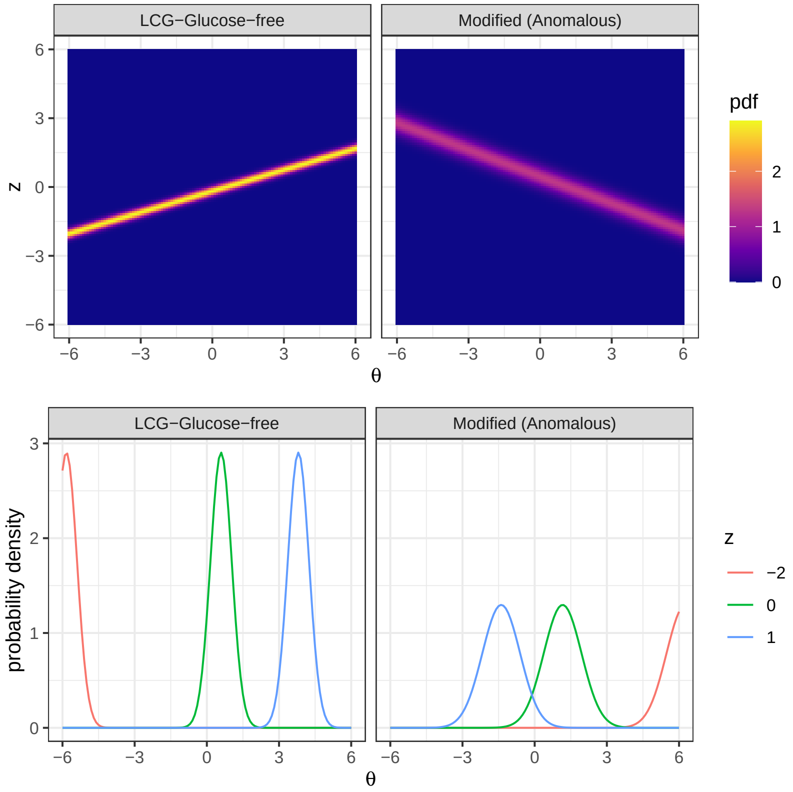


Figure 11: The left column shows the heatmap and the trace lines for LCG-Glucose-free, a typical algorithm with increasing θ corresponding to increased performance. The right column shows the heatmap and the trace lines for an anomalous algorithm, which obtains high accuracy scores for difficult test instances and low accuracy scores for easy test instances.

3.3 Algorithm metric: Anomalous indicator

Consider the heatmap and the trace lines shown in Figure 11. The left column represents the algorithm LCG-Glucose-free and the second column represents a different type of algorithm. The second algorithm is constructed using an algorithm in the Minizinc portfolio for illustrative purposes. Suppose these figures were generated from an item in educational testing. Then the left column shows the heatmap and the trace lines of a test item for which higher examinee ability corresponds to higher grades. On the other hand the right column shows a test item for which examinees with lower ability obtain higher grades than examinees with higher ability. Such a test item is said to have negative discrimination. The standard premise in educational testing is that high grades correspond to high ability. As such, a test item with negative discrimination is commonly revised to obtain a positive discrimination or removed from the pool of questions (Hambleton and Swaminathan, 2013).

However, in algorithm evaluation such a heatmap or a set of trace lines represent an algorithm or a dataset with an interesting quirk. For the standard IRT approach for algorithm evaluation, the heatmap and the trace lines represent a dataset, which gives poor performances for high ability algorithms and good performances for low ability algorithms. For the inverted IRT approach, the heatmap and trace lines represent an algorithm that performs well on difficult test instances and poorly on easy test instances. We describe such algorithms as "anomalous". Indeed, the *no free lunch* concept emphasizes that no single algorithm performs better than other algorithms for all problems.

This is confirmed by the *instance space* analyses conducted by Smith-Miles and co-authors (Smith-Miles and Tan, 2012; Kang et al., 2017; Muñoz and Smith-Miles, 2017). Furthermore, the instance space analyses for different problems show that even though some algorithms perform poorly on average, they often hold a niche in the instance space where they outperform other algorithms (Kandanaarachchi et al., 2019). As this is a unique strength of the algorithm, it should not be removed from the dataset as practiced in educational testing.

For continuous and polytomous IRT models, the standard parameters for item j comprise the discrimination parameter α_j and the difficulty parameter β_j for continuous models, and the intercepts $\boldsymbol{d}_j = \left(d_1, \ldots, d_{C_j-1}\right)$ for polytomous models. The discrimination parameter, which is present in both continuous and polytomous models highlights two aspects of algorithm performance. The sign of the discrimination parameter tells us if the algorithm is typical or anomalous. If $a_j < 0$ then algorithm j gives better performance values for difficult test instances and low performances for easy test instances, and is considered anomalous. So we define the anomalous indicator as

$$\text{anomalous}(j) = \begin{cases} \text{TRUE} & a_j < 0\,, \\ \text{FALSE} & \text{otherwise}\,. \end{cases}$$

3.4 Algorithm metric: Algorithm consistency score

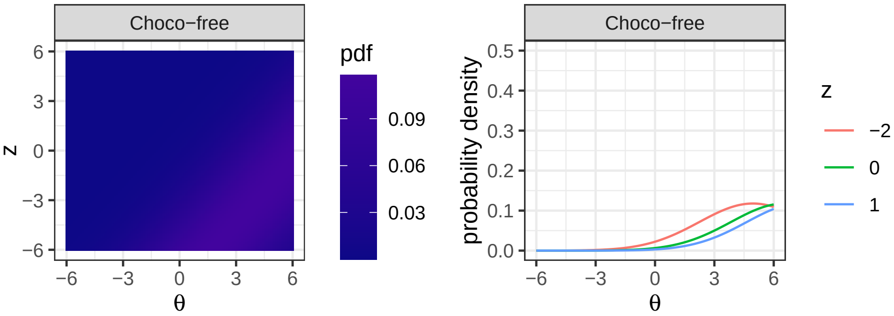


Figure 12: The heatmap and the trace lines for Choco-free, a relatively consistent algorithm in this portfolio.

Consider the heatmap and the trace lines in Figure 12. Suppose these trace lines relate to a test item in an educational testing scenario. Then this item does a poor job in discriminating examinees with different abilities, because all examinees are most likely to obtain a similar score regardless of their ability.

In algorithm evaluation using the standard IRT approach, such a heatmap and trace lines indicate that the dataset in question does not discriminate between the algorithms. That is, the dataset might be too difficult for all algorithms or too easy for all algorithms. Similarly, in algorithm evaluation using the inverted IRT approach, such a heatmap and trace lines indicate that the algorithm does not discriminate. That is, regardless of the easiness/difficulty of the test instance, this algorithm is most likely to give a similar score, i.e. its sensitivity to test instances is quite low. Thus, the algorithm is consistent and non-discriminative. The consistency or robustness of an algorithm is an important characteristic that is sometimes overlooked in the quest for peak performance.

Stability or robustness can be defined in different ways. For example, Eiben and Smit (2011) discuss 3 types of robustness indicators: robustness with respect to parameters, problem specification and random seeds. Often robustness or stability is defined as a measure of the change of the output with respect to a small perturbation of the input. In our case, we do not perturb the input; however datasets positioned close to each other in the latent trait continuum are considered to have similar easiness/difficulty level. As such, a measure of the change of performance values across the latent trait continuum is an indication of stability or robustness. However, stability or robustness are positive attributes. The algorithm quality we want to encapsulate is slightly different in the sense that some algorithms can consistently perform poorly irrespective of the problem while others can consistently perform well. We capture this notion by defining algorithm consistency.

The absolute value of the discrimination parameter $|a_j|$ gives the discrimination power of the algorithm, which is linked to the consistency of the algorithm. If $|a_j|$ is small, then the algorithm will produce trace curves with slower transitions similar to those in Figure 12, signifying a more consistent algorithm than one with a larger $|a_j|$. As such, we define consistency as

$$\text{consistency}(j) = \frac{1}{|a_j|} \, .$$

Tying this back to the heatmaps, the discrimination power of the algorithm is connected with the sharpness of the lines/bands on the heat map. In Figure 10 we see that some algorithms have sharp lines while others have blurry lines. Algorithms with sharp lines are more discriminating than algorithms with blurry lines, i.e., algorithms with blurry lines or no lines are more consistent than algorithms with sharp lines.

3.5 Algorithm metric: Difficulty limit

Both consistency and anomalousness relate to the IRT discrimination parameter. Next, we discuss the role of the item difficulty parameter in the inverted IRT algorithm evaluation approach. Suppose Figure 13 represents two items in educational testing. The first and the second columns in Figure 13 show the trace lines and the heatmaps of two items, with the item in the left column having higher difficulty. We see that for any given ability θ, the most probable score in the heatmap in the right column is higher than that of the left column.

In the inverted IRT approach, the heatmaps and the tracelines represent algorithms with the algorithm in the left column, Mistral-free, giving lower performance for similar datasets compared to the algorithm in the right column, MZN/SCIP-free. When we consider dataset difficulty ($-\theta$), we see that as datasets get more difficult the algorithm performance goes down. Thus, each algorithm has an upper limit in terms of dataset difficulty. If the difficulty of a dataset is lower than this limit, we expect the algorithm to give good results, but if it is higher than the limit, the algorithm would

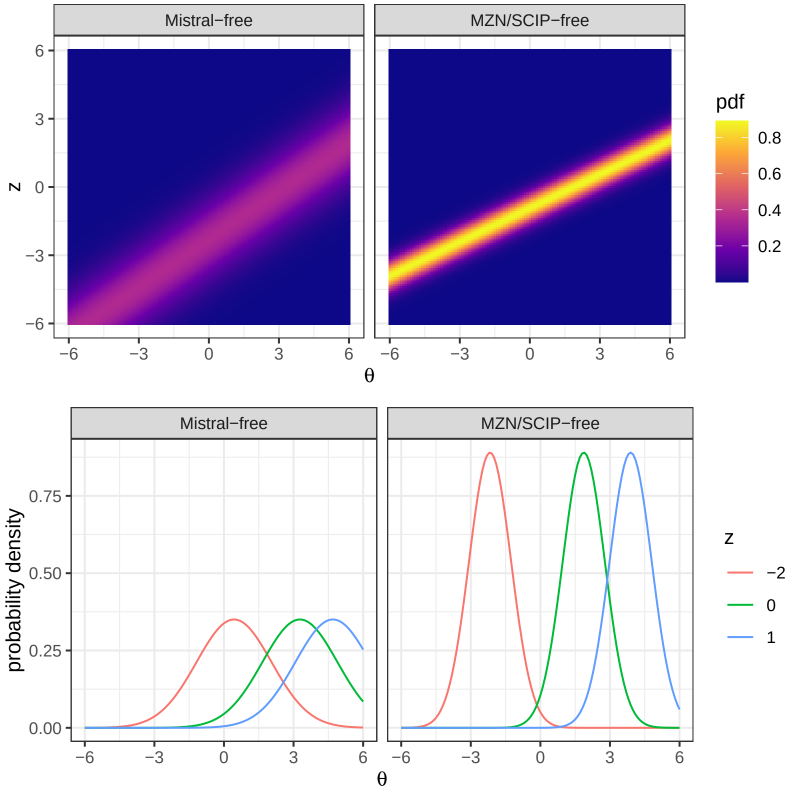


Figure 13: Two algorithms with different difficulty limits. Mistral-free has a higher difficulty limit than MZN/SCIP-free.

perform poorly. Therefore, we define the algorithm difficulty limit as

$$\text{difficulty}(j) = -\beta_j \,,$$

where β_j is the traditional IRT difficulty parameter. Higher values of difficulty(j) indicate better algorithms that can handle more difficult datasets.

For polytomous IRT, as there are multiple difficulty parameters $(d_1, d_2, \ldots, d_{C_j-1})$, we use $-d_{C_j-1}$ as the difficulty limit, because this denotes the threshold for the highest performance level.

4. Evaluating algorithm portfolios using AIRT

4.1 Modelling algorithm performance based on dataset difficulty

The dataset difficulty spectrum gives a way of ordering the performance values y_{ij}. For each algorithm j, we can consider the set of points (δ_i, y_{ij}) for $i \in \{1, \ldots, N\}$. When ordered by δ_i, y_{ij} exhibits algorithm j's performance as datasets get progressively difficult. Thus, for each algorithm j, we can fit a model explaining the performance by dataset difficulty values. These models can be denoted by functions $\{h_j(\delta)\}_{j=1}^n$, where j denotes the algorithm and δ the dataset difficulty. For simplicity our h_j's are smoothing splines.

The smoothing spline h_j minimizes the function

$$\sum_{i=1}^{N} \left(y_{ij} - h_j(\delta_i)\right)^2 + \lambda \int h_j''(t)\, dt ,$$

where the first term denotes the sum of squared errors and the second term is a penalty for wiggliness. It is the second term – the integral of the second derivative – that gives the smoothness to the spline. The parameter λ is a tuning parameter and is computed by using a closed-form expression that minimizes the leave-one-out cross validation squared error (James et al., 2013).

An advantage of using smoothing splines is that we do not need to specify any parameters to fit the splines. Furthermore, by graphing the splines we can visualize regions of the latent trait where algorithms give good or weak performance.

We note that this is a feature-less way of exploring algorithm performance. For example, in instance space analysis we compute features of datasets and explain algorithm performance using these features. AIRT explains algorithm performance using dataset difficulty, which is computed from fitting an IRT model without using external features.

CSP-Minizinc-2016 algorithm portfolio ordered by dataset difficulty and the fitted smoothing splines are shown in Figure 14. From this diagram we see that different algorithms perform better for different values of dataset difficulty.

4.2 Strengths and weaknesses of algorithms

We can compute the strengths and weaknesses of algorithms using the dataset/problem difficulty spectrum. To find the algorithm strengths we first find the best algorithm performance for each value δ in the problem difficulty spectrum. That is,

$$h_{j_*}(\delta) = \max_j h_j(\delta) .$$

Next, for a given $\epsilon > 0$ we define the strengths of algorithm j as

$$\text{strengths}(j, \epsilon) = \left\{ \delta : |h_j(\delta) - h_{j_*}(\delta)| \leq \epsilon \right\} .$$

That is, the strengths of algorithm j denote the regions in the problem difficulty spectrum where algorithm j gives good performance. Here good is defined as close to best, specifically within ϵ from the best. As such, we can get multiple contiguous regions of strengths for some algorithms while others may not have any strengths in the spectrum for a given ϵ.

Algorithm weaknesses are found similarly. To compute the weaknesses we first find the poorest algorithm performance for every point in the problem difficulty spectrum:

$$h_{j_\#}(\delta) = \min_j h_j(\delta) .$$

Then, we define the weaknesses of algorithm j as

$$\text{weaknesses}(j, \epsilon) = \left\{ \delta : |h_j(\delta) - h_{j_\#}(\delta)| \leq \epsilon \right\} .$$

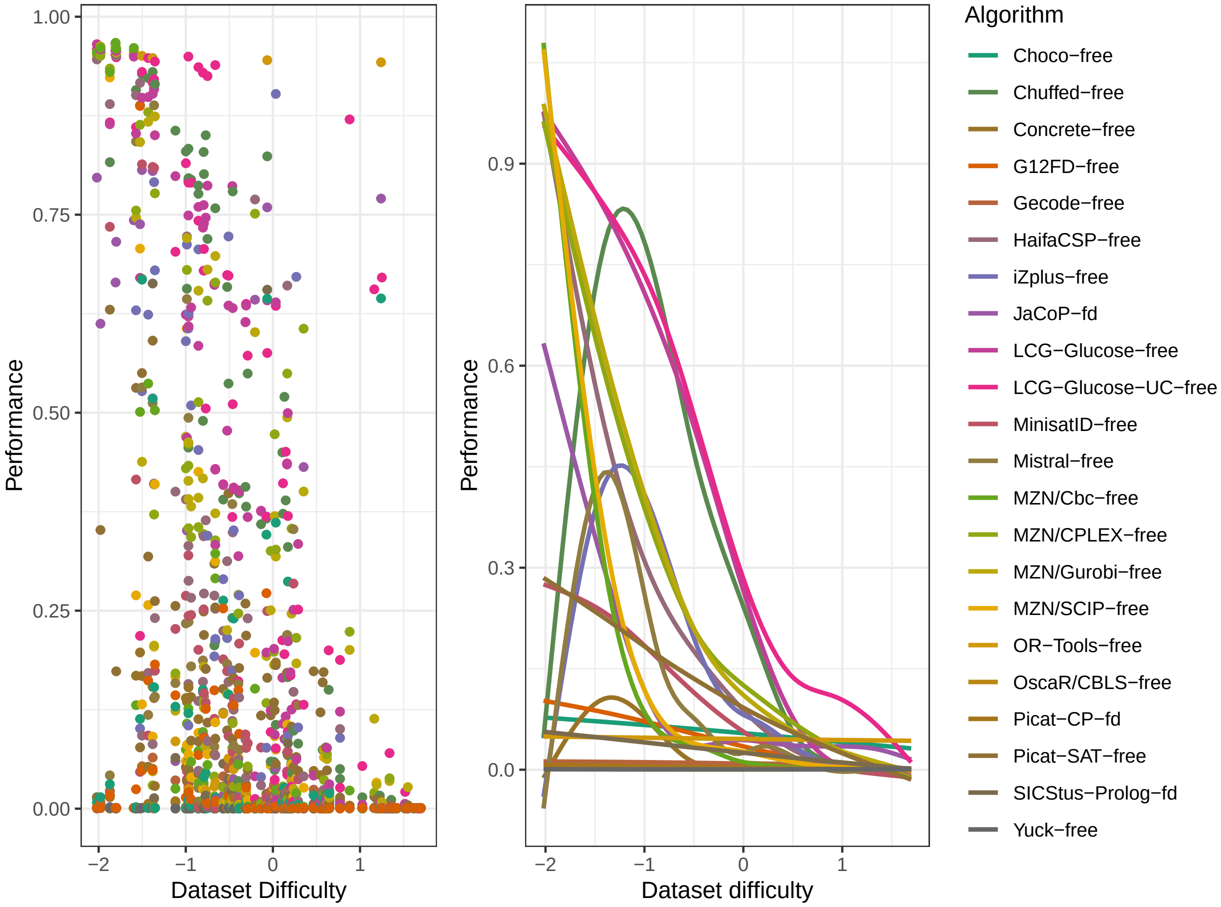


Figure 14: The dataset difficulty spectrum of CSP-Minizinc-2016 explored. Left: Algorithm performance against dataset difficulty for all 4 algorithms. Right: Smoothing splines fitted to algorithm performance values.

Weaknesses represent regions in the problem difficulty spectrum where algorithms give poor performance.

Figure 15 shows the strengths and weaknesses of CSP-Mnizinc-2016 algorithm portfolio. The top row shows the strengths and weaknesses for $\epsilon = 0$ and the bottom part for $\epsilon = 0.01$. The difference between the two values of ϵ is that when $\epsilon = 0$, for each value δ in the dataset difficulty spectrum there is only one algorithm that is strong. When $\epsilon \neq 0$ multiple algorithms can display strengths for the same δ.

In Figure 15 we see that when $\epsilon = 0$ LCG-Glucose-UC-free is strong for a large part of the problem space, including difficult and medium-difficult problems. OR-Tools-free is better for more difficult problems and LCG-Glucose-free and Chuffed-free for easy problems. For $\epsilon = 0$ only 5 algorithms have strengths. When $\epsilon = 0.01$ we see a little overlap. However, when $\epsilon = 0.01$ only 7 algorithms out of 22 algorithms exhibit strengths. In contrast, 16 algorithms have weaknesses when $\epsilon = 0.01$. Both LCG-Glucose-UC-free and LCG-Glucose-free have strengths for easier problems but LCG-Glucose-UC-free remains the more powerful algorithm. In the weaknesses space, we see Picat-CP-fd, OscaR/CBLS-free and Yuck-free displaying weaknesses for most of the problem space. A large number of algorithms are weak for difficult problems as seen for $\epsilon = 0.01$.

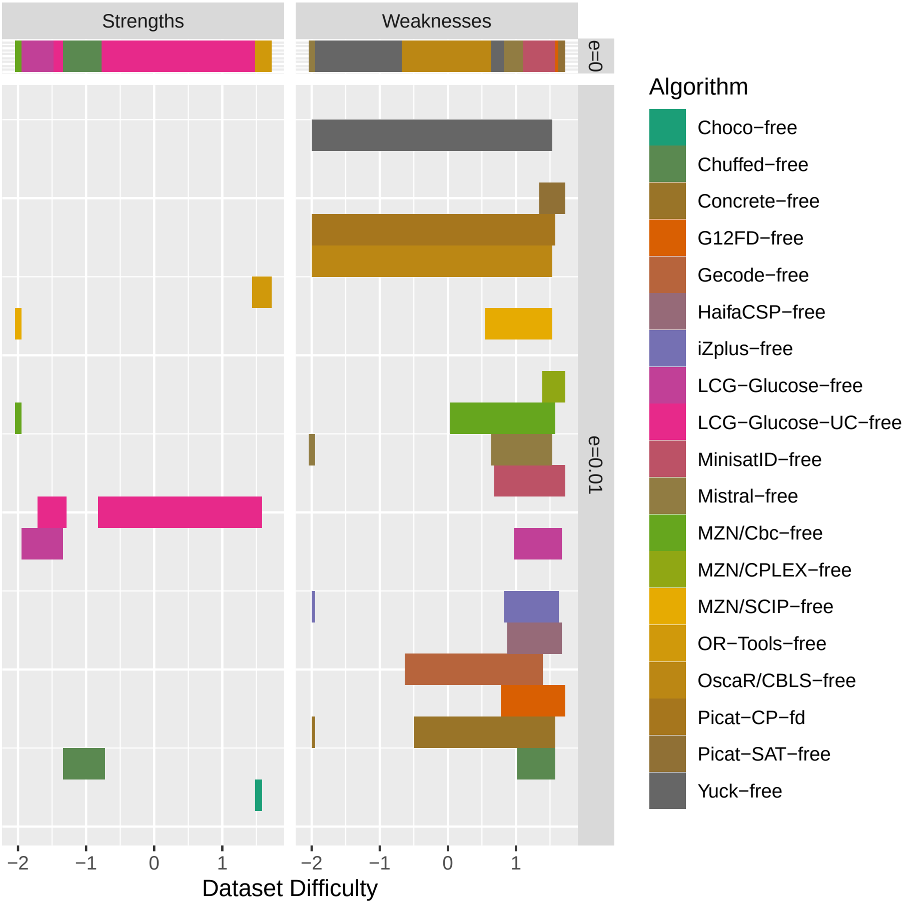


Figure 15: Strengths and weaknesses of CSP-Minizinc-2016 algorithms for $\epsilon = 0$ and $\epsilon = 0.01$.

Using the strengths we compute the *latent trait occupancy* (LTO) for each algorithm. LTO gives the proportion of datasets supported by each algorithm in the region of its strength. We define it as

$$\text{LTO}(j, \epsilon) = \frac{|\{i : i \in \text{strengths}(j, \epsilon)\}|}{N} ,$$

where i and j denote the datasets and algorithms respectively. The total number of datasets/problems is denoted by N. For the strengths shown in Figure 15 for $\epsilon = 0$, LCG-Glucose-UC-free occupies the largest portion of the latent trait followed by Chuffed-free. When $\epsilon > 0$, the quantity $\sum \text{LTO} > 1$ if the strengths of algorithms overlap as shown in Figure 15. The LTO values for both $\epsilon = 0$ and $\epsilon = 0.01$ is listed in Table 2.

Combining Figure 15 and Table 2 we see that for very easy problems ($\delta \approx -2$) algorithms MZN/Cbc-free and MZN/SCIP-free display strengths. However, we see that the latent trait occupancy LTO = 0.02, which is very small. Therefore, even if these two algorithms have strengths for very easy problems, it is risky to use them because of small LTO. For easy problems ($\delta \leq 0$) we have 3 candidates: LCG-Glucose-UC-free, Chuffed-free and LCG-Glucose-free. The LTO of these algorithms are 0.828, 0.141 and 0.111 respectively. Basically, this reiterates that LCG-Glucose-UC-free is the most powerful algorithm. For very hard datasets ($\delta > 1$), we have 3 candidates, Choco-free, OR-Tools-free and LCG-Glucose-UC-free. Of these, Choco-free has an LTO of 0.04, and thus can be disregarded. OR-Tools-free occupies the same position in the strengths diagram for both $\epsilon = 0$ and $\epsilon = 0.01$ and thus has a unique strength for very difficult problems.

Table 2: AIRT Latent Trait Occupancy (LTO) for CSP-Minizinc-2016 algorithms.

Algorithm	LTO ($\epsilon = 0$)	LTO ($\epsilon = 0.01$)
LCG-Glucose-UC-free	0.717	0.828
Chuffed-free	0.121	0.141
LCG-Glucose-free	0.071	0.111
OR-Tools-free	0.071	0.071
MZN/Cbc-free	0.020	0.020
Choco-free	0	0.040
MZN/SCIP-free	0	0.020

4.3 Algorithm portfolio selection

The analysis in the previous section can be used understand the strengths and weaknesses of algorithms, adding to the exploratory data analysis domain of algorithm portfolios. We can also use AIRT for algorithm portfolio selection. We construct the airt portfolio by selecting the set of strong algorithms for a given ϵ.

Formally, the airt portfolio is defined as

$$\begin{aligned}\mathcal{A}(\epsilon) &= \left\{ j : |h_j(\delta) - h_{j_*}(\delta)| \leq \epsilon, \text{ for all } \delta \right\} , \\ &= \{ j : \text{strengths}(j, \epsilon) \neq \emptyset \} .\end{aligned}$$

When $\epsilon = 0$ we obtain $\mathcal{A}(0) = \bigcup j_*$, i.e., the strongest set of algorithms in the latent space.

We use lowercase letters ‘airt’ when describing portfolio specific results and uppercase AIRT when describing more general aspects. The number of algorithms in the airt portfolio depends on ϵ. However, we do not directly specify the number of algorithms. It is a result of the smoothing splines $\{h_j(\delta)\}_{j=1}^{n}$, which use the dataset difficulty spectrum δ as the input. But, $\delta_i = -\theta_i$, which is computed using α_j, β_j, γ_j and z_{ij} as dictated by equation (10). Therefore, the AIRT model has a direct influence on the portfolio.

Of course, the airt portfolio, strengths and weaknesses and other indicators of algorithm performance are only reliable if the IRT model providing the parameters has a good fit. In the following section we provide some measures of goodness of the IRT model to support interpretation of the results.

4.4 IRT Model goodness measures

We are using IRT to model algorithm performance, that is the IRT model is effectively a meta-model. Checking the accuracy or the goodness of the IRT model is important because it determines the confidence we can place on the IRT model parameters, which describe the algorithms. If the IRT model is accurate, then we can trust the relationships it has modeled between instances and algorithm performances.

After fitting a continuous (polytomous) IRT model we define the predicted result (category) for a test instance i, with latent score θ_i as the result (category) with the highest probability for latent score θ_i. We denote the predicted result (category) for test instance i and algorithm j by $\hat{x}_{ij}$. Then the residuals $e_{ij} = x_{ij} - \hat{x}_{ij}$ are of interest to us. For a fixed j, let $e_j = \{e_{ij}\}_{i=1}^{N}$ denote the residuals of the j^{th} algorithm. We consider the scaled absolute residuals $\rho_{ij} = c|e_{ij}|$, such that $\rho_{ij} \in [0, 1]$. As we are interested in the algorithms we define $\rho_j = \{\rho_{ij}\}_{i=1}^{N}$ and consider the empirical cumulative distribution function (CDF) of ρ_j for each j, which we denote by $F(\rho_j)$:

$$F(\rho_j) = P(\rho_j \leq \rho) \quad \text{for} \quad \rho \in [0, 1] \,. \tag{12}$$

Figure 16 shows a histogram of the absolute residuals $|e_{ij}|$, the empirical cumulative distribution functions of $|e_{ij}|$, and the scaled absolute residuals ρ_{ij} for iZplus-free algorithm in CSP-Minizinc-2016 portfolio. The only difference between the two CDFs is the x values, which are in the interval $[0, 1]$ for the scaled absolute residuals.

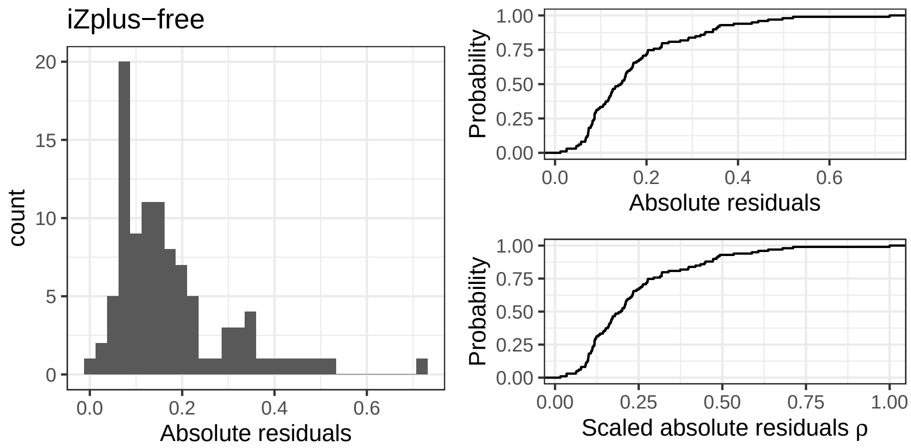


Figure 16: The histogram of the absolute residuals $|e_{ij}|$ of iZplus-free is shown on the left. The CDF of the absolute residuals is shown on the top right and the CDF of the scaled absolute residuals is shown on the bottom right. Notice the difference in the domain for the two CDFs.

By rescaling the absolute residuals to $[0, 1]$ we make sure that the area under the CDF $F(\rho_j)$, denoted by AUCDF(ρ_j), is bounded by 1. AUCDF(ρ_j) provides a measure of goodness of the IRT model for algorithm j. A higher AUCDF signifies better IRT model fit.

We compute the mean square error (MSE) of the residuals and AUCDF(ρ_j) for each algorithm j. IRT may fit some algorithms better than others. We note that the log-likelihood obtained from fitting the IRT model is an aggregate and therefore does not show how well each algorithm is fitted.

By computing the residual metrics such as mean square error and AUCDF(ρ_j) we gain a better understanding of the IRT model in relation to each algorithm.

4.5 Predicted and actual effectiveness

We are interested in how well algorithms perform on test instances, especially the high performance results. If an algorithm gives good performance results for most test instances, then that algorithm is effective. As such, we focus on the performance results in decreasing order and study effectiveness via the cumulative distribution function (CDF) for each algorithm. First we denote the algorithm performance results for algorithm j by $x_j = \{x_{ij}\}_{i=1}^N$. By defining $t_j = \max(x_j) - x_j$ we reverse the performance results so that small values of t_j denote high performance results. The variable t_j can be thought of as a tolerance parameter, i.e. small tolerances give better performance. Then we compute the effectiveness of the algorithm by

$$\bar{F}_j(\ell) = P(t_j \leq \ell),$$

where P denotes the probability. The function $\bar{F}_j(\ell)$ is also related to the complementary cumulative distribution (CCDF), which is defined as

$$\bar{F}_x(\ell) = P(x \geq \ell),$$

since,

$$\begin{aligned} P(t_j \leq \ell) &= P\left(\max(x_j) - x_j \leq \ell\right) , \\ &= P\left(x_j \geq \max(x_j) - \ell\right) . \end{aligned}$$

As such, $\bar{F}_j(\ell)$ denotes the CCDF of x_j with the x axis reversed.

We call the curve $y = \bar{F}_j(\ell)$, the effectiveness curve. By scaling t_j to lie in $[0, 1]$, we make sure that the area under the effectiveness curve is bounded by 1. For polytomous IRT with categories $\{0, 1, \dots, C_{j-1}\}$, we consider a step size of $\Delta = \frac{1}{C_{j-1}}$ for the x axis with $\ell \in \{0, 1, \dots, C_{j-1}\}$, so that the curve $y = \bar{F}_j(\ell)$ is defined by the points $\left(0, \bar{F}_j(C_{j-1})\right), \left(\Delta, \bar{F}_j(C_{j-2})\right), \dots, \left(1, \bar{F}_j(0)\right)$. Figure 17 shows the histogram, CDF of performance values (bottom-left) and the effectiveness curve (bottom-right) of Chuffed-free algorithm in CSP-Minizinc-2016 portfolio.

Similarly, we can compute the effectiveness for the IRT predicted algorithm performance values by defining $\hat{x}_j = \{\hat{x}_{ij}\}_{i=1}^N$, and $\hat{t}_j = \max \hat{x}_j - \hat{x}_j$ where $\hat{x}_{ij}$ denotes the predicted result for algorithm j and test instance i. This gives the predicted effectiveness

$$\bar{F}_j(\hat{\ell}) = P(\hat{t}_j \leq \ell),$$

where we have indicated that it is a predicted quantity by using $\hat{\ell}$. We have denoted the effectiveness by $\bar{F}$ for both predicted and actual values, while changing from ℓ to $\hat{\ell}$ for predicted effectiveness. We compute the area under the actual and predicted effectiveness curves as this is a measure of an algorithm's ability to produce high performance results. We denote the area under the actual effectiveness curve $y = \bar{F}_j(\ell)$ by AUAEC(j), and area under the predicted effectiveness curve $y = \bar{F}_j(\hat{\ell})$ by AUPEC(j). A high AUAEC(j) indicates that algorithm j has a large proportion of high

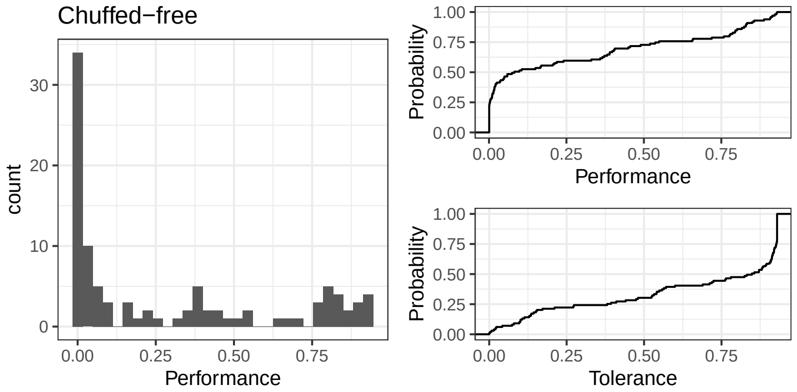


Figure 17: The histogram of performance values Chuffed-free algorithm is shown on the left. The graph on the top right shows the CDF of the performance values. The graph on the bottom right shows the effectiveness curve $y = \bar{F}_j(\ell)$.

performance results and a high AUPEC(j) indicates that the IRT model predicts algorithm j to have a large proportion of high performance results.

For a single algorithm the pair of values (AUAEC, AUPEC) gives an indication about the algorithm's actual and perceived ability to produce high performance results. If the absolute difference between the predicted and actual effectiveness, |AUAEC − AUPEC| is large, then the trustworthiness of the IRT model is low for that algorithm. It may be the case that AUAEC ≈ AUPEC for most algorithms in a portfolio, but for one algorithm the absolute difference between AUAEC and AUPEC is higher. A larger absolute difference between AUAEC and AUPEC will concur with a lower AUCDF for that algorithm. For example if the IRT model over estimates the performance of an algorithm, AUPEC will be higher than AUAEC. This will also result in lower agreement between the predicted and the actual results giving rise to lower AUCDF. Table 3 gives the model goodness measures for each algorithm. We see that the MSE is low for most algorithms apart from HaifaCSP-free, which also has the highest |AUAEC − AUPEC|. In terms of goodness of fit, we can say that the IRT model is a good fit for mostly all algorithms, apart from HaifaCSP-free.

The measures we have proposed for algorithm consistency score, anomalous indicator and difficulty limit are algorithm evaluation metrics while the absolute residuals curve, actual and predicted effectiveness curves, along with AUCDF, |AUAEC - AUPEC| comprise AIRT's model goodness metrics. In the discussion that follows we refer to both AIRT and the underlying IRT model. AIRT refers to the reinterpreted IRT model with the additional evaluation metrics discussed above. When we discuss standard IRT concepts such as trace lines we refer to the IRT model.

This concludes the discussion on different aspects of the AIRT framework. The pseudocode given in Algorithm 1 summarizes the steps and functionality of AIRT.

4.6 Computational complexity of AIRT

To fit the IRT model, we use the non-iterative item parameter solution proposed by Shojima (2005). They use expectation maximization (EM) and in each EM cycle a non-iterative solution is found by optimizing the expectation in equation (9) item-by-item. By computing partial derivatives and

Algorithm 1: *AIRT framework.*

input : The matrix $Y_{N\times n}$, containing accuracy measures of n algorithms for N datasets/problem instances.

output : 1. AIRT indicators of algorithms and dataset/problem difficulty
2. The strengths and weaknesses of algorithms
3. airt algorithm portfolio
4. Model goodness measures

Stage 1 - Fitting the IRT model with inverted mapping

1. Transform the accuracy measures y_{ij} by defining $z_{ij} = \ln \frac{y_{ij}}{k-y_{ij}}$.

2. Let $Z = \{z_{ij}\} \in \mathbb{R}^{N\times n}$, where N denotes the number of problems/datasets and n denotes the number of algorithms.

3. Fit a continuous IRT model to Z by maximizing the log-likelihood function

$$E_{\boldsymbol{\theta}|\Lambda^{(t)},\boldsymbol{Z}}\left[\ln p\left(\boldsymbol{\Lambda}|\boldsymbol{\theta},\boldsymbol{Z}\right)\right] = N\sum_{j=1}^{n}\left(\ln|\alpha_j| + \ln|\gamma_j|\right) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{n}\alpha_j^2\left(\left(\beta_j + \gamma_j z_{ij} - \mu_i^{(t)}\right)^2 + \sigma^{(t)2}\right) + \ln p\left(\boldsymbol{\Lambda}\right) + \text{const},$$

4. From this model we obtain (after t iterations) the IRT discrimination and difficulty parameters α_j and β_j and the scaling parameter γ_j for algorithms $j \in \{1,\ldots,n\}$ as follows:

$$\begin{aligned}
\gamma_j^{(t+1)} &= \frac{V\left(\mu_i^{(t)}\right) + \sigma^{(t)2}}{C_j\left(z_{ij}, \mu_i^{(t)}\right)}, \\
\beta_j^{(t+1)} &= M\left(\mu_i^{(t)}\right) - \gamma_j^{(t+1)} M_j\left(z_{ij}\right), \\
\alpha_j^{(t+1)} &= \operatorname{sign}\left(\gamma_j^{(t+1)}\right)\left(\gamma_j^{(t+1)2} V_j(z_{ij}) - V\left(\mu_i^{(t)}\right) - \sigma^{(t)2}\right)^{-1/2}.
\end{aligned}$$

5. Using these IRT parameters we compute the latent trait $\theta_{N\times 1}$ as

$$\theta_i = \frac{\sum_j \hat{\alpha}_j^2\left(\hat{\beta}_j + \hat{\gamma}_j z_{ij}\right)}{\sum_j \hat{\alpha}_j^2}.$$

Stage 2 - Calculation of algorithm and dataset metrics

6. For each algorithm j compute the anomalous indicator, algorithm consistency score and difficulty limit using

$$\begin{aligned}
\text{anomalous}(j) &= \begin{cases} \text{TRUE} & a_j < 0, \\ \text{FALSE} & \text{otherwise}. \end{cases}, \\
\text{consistency}(j) &= \frac{1}{|a_j|}, \\
\text{difficulty}(j) &= -\beta_j,
\end{aligned}$$

7. For each dataset i compute the dataset difficulty using
$\delta_i = -\theta_i$.

Stage 3 - Computing strengths and weaknesses and construct airt portfolio

8. Using the dataset difficulty spectrum δ fit smoothing splines $h_j(\delta)$ to performance values y_{ij} for each algorithm j minimizing $\sum_{i=1}^{N}\left(y_{ij} - h_j(\delta_i)\right)^2 + \lambda\int h_j''(t)\,dt$.

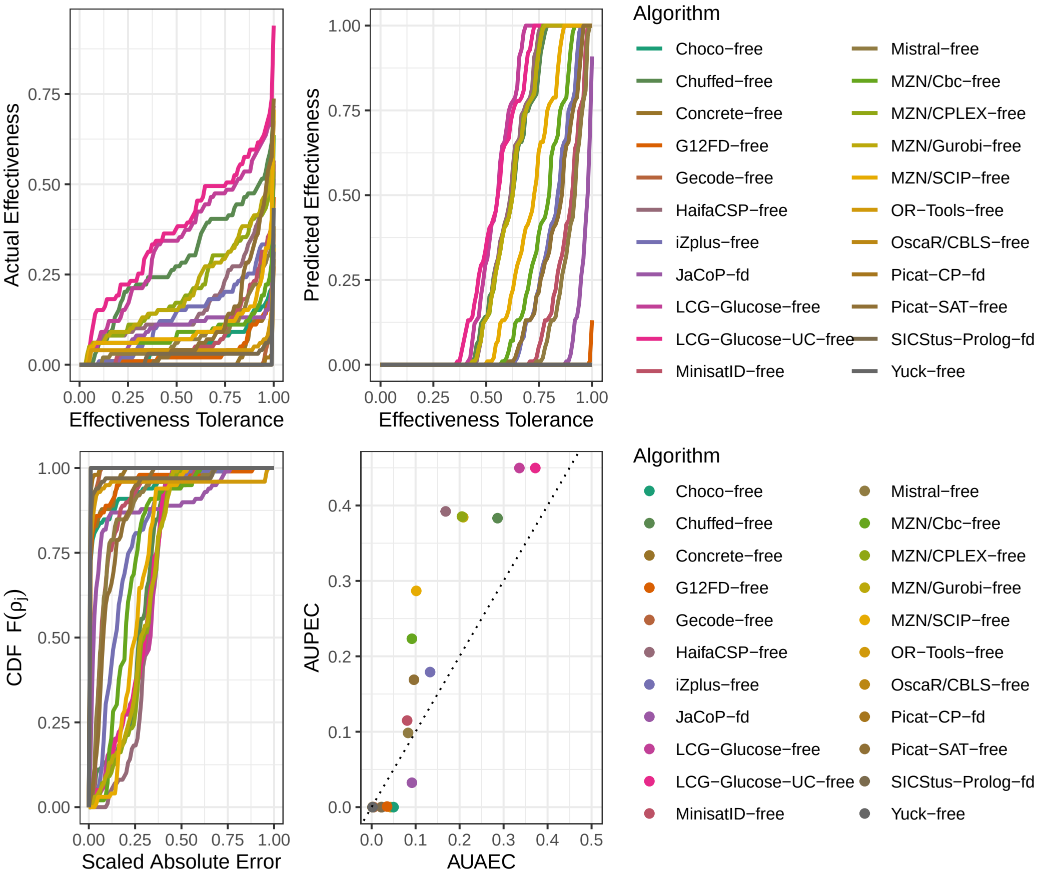


Figure 18: The model goodness graphs for CSP-Minizinc-2016 portfolio. The top row shows actual and predicted effectiveness curves. The graph on the bottom left shows the CDF of the absolute residuals and the graph on the bottom right shows the actual and predicted effectiveness of the algorithms.

9. Compute the strengths and weaknesses of algorithms using

$$\text{strengths}(j, \epsilon) = \left\{ \delta : |h_j(\delta) - h_{j_*}(\delta)| \leq \epsilon \right\} ,$$
$$\text{weaknesses}(j, \epsilon) = \left\{ \delta : |h_j(\delta) - h_{j_\#}(\delta)| \leq \epsilon \right\} ,$$

and use the strengths and weaknesses for exploratory data analysis purposes.
10. Construct the airt portfolio using $\mathcal{A}(\epsilon) = \{ j : \text{strengths}(j, \epsilon) \neq \emptyset \}$.
11. Check the fit of the IRT model by computing model goodness measures MSE, AUCDF and |AUAEC - AUPEC|.

solving a set of simultaneous equations they find the exact solutions for item parameters α_j, β_j and γ_j in each cycle. The optimization stops when solutions of successive cycles converge or when the maximum number of cycles is reached. Let c denote the number of cycles. Hence, the computation is repeated c times. For a $N \times n$ matrix Z, there are n items and N participants. The

Table 3: MSE, AUCDF, Area Under Actual Effectiveness Curve (AUAEC) and Predicted Effectiveness Curves (AUPEC) and |AUAEC - AUPEC| for CSP-Minizinc algorithms.

Algorithm	MSE	AUCDF	AUAEC	AUPEC	|AUAEC - AUPEC|
iZplus-free	0.046	0.823	0.133	0.179	0.046
MZN/SCIP-free	0.072	0.746	0.102	0.287	0.185
Chuffed-free	0.085	0.733	0.286	0.383	0.097
LCG-Glucose-UC-free	0.088	0.725	0.372	0.450	0.078
Concrete-free	0.003	0.974	0.022	0.000	0.022
JaCoP-fd	0.051	0.894	0.092	0.032	0.059
Mistral-free	0.027	0.892	0.083	0.098	0.016
OscaR/CBLS-free	0.000	0.995	0.002	0.000	0.002
HaifaCSP-free	0.100	0.690	0.168	0.392	0.224
Gecode-free	0.000	0.989	0.005	0.000	0.005
OR-Tools-free	0.037	0.951	0.043	0.000	0.043
SICStus-Prolog-fd	0.013	0.972	0.023	0.000	0.023
Picat-CP-fd	0.000	0.993	0.002	0.000	0.002
Picat-SAT-free	0.017	0.894	0.096	0.169	0.073
MZN/Gurobi-free	0.089	0.720	0.208	0.384	0.176
MZN/CPLEX-free	0.093	0.713	0.205	0.385	0.180
LCG-Glucose-free	0.089	0.722	0.336	0.450	0.114
MZN/Cbc-free	0.058	0.786	0.092	0.223	0.132
Yuck-free	0.000	0.995	0.002	0.000	0.002
Choco-free	0.021	0.944	0.050	0.000	0.050
MinisatID-free	0.022	0.898	0.081	0.115	0.034
G12FD-free	0.014	0.961	0.035	0.001	0.034

non-iterative solution is found for each item $j \in \{1, \dots, n\}$. For a fixed j, solving for α_j, β_j and γ_j involves computing various quantities such as mean, variance and covariance. The computational complexity of these operations is $\mathcal{O}(N)$. When they are computed for each item j for c cycles the overall complexity of fitting the IRT model becomes $\mathcal{O}(Nnc)$. Of the three variables c has an upper bound of 200 and n is much smaller than N. As such the most influencing variable is N.

After fitting the IRT model we compute the anomalous indicator, algorithm consistency score and difficulty limit for each algorithm. These computations take a fixed amount of time for each algorithm j. Therefore, computing the indicators have $\mathcal{O}(n)$ complexity. Computing dataset difficulty values δ_i for N datasets using equations (10) and (11) takes $\mathcal{O}(N)$ complexity. Therefore, computing AIRT indicators and dataset difficulty values have $\mathcal{O}(N + n) \approx \mathcal{O}(N)$ complexity as n is much smaller compared to N.

Smoothing splines can be fitted in $\mathcal{O}(N)$ computational time. In statistical software packages, they are fitted using a much smaller number of points, approximately $\log(N)$ when $N > 50$ (Hastie et al., 2009). Strengths and weaknesses of algorithms are computed mainly for visualization purposes. As such, the strengths and weaknesses horizontal bar graph has a smaller number of points compared to N; let us say it has M points. For each of these points we compute the strengths and weaknesses of n algorithms. This computation involves $O(Mn)$ complexity; however vectorized computations make it much faster. The airt algorithm portfolio can be computed in fixed time as they take the union of strong or weak algorithms.

Model goodness measures involve n algorithms with N data points for each algorithm. Computing the MSE, and the CDF for ρ_j as in equation (12) have $O(nN)$ complexity. Computing the area under the curve using trapezoidal integration takes $O(N)$ time. Similarly, actual and predicted effectiveness have $O(nN)$ complexity.

5. Results

We now test AIRT on 10 algorithm portfolios hosted on ASlib data repository (Bischl et al., 2016). ASlib hosts performance data and test instance features for a large number of algorithm portfolios. Section 5.1 contains a detailed analysis of classification algorithms using AIRT. We explore AIRT metrics, model goodness measures and the strengths and weakness of algorithms using the dataset difficulty spectrum. In addition, we compare different algorithm portfolios. The analysis of classification algorithms encompasses the full functionality of AIRT. We carry out more concise analyses for other ASlib scenarios in Appendix A. We include the latent trait curves, strengths and weaknesses and algorithm portfolio comparisons for each ASlib scenario.

5.1 Detailed case study: Classification

This scenario was introduced by van Rijn (2016) and uses a selection of WEKA algorithms (Hall et al., 2009). It was later used in the 2017 algorithm selection challenge by Lindauer et al. (2017). The dataset contains predictive accuracy results from 30 classification algorithms on 105 test instances. The default parameters and hyperparameters used by the classification algorithms were not varied. For ease of plotting graphs, we have shortened the names of many algorithms. For example, there are 3 multilayer perceptron algorithms; algorithm 8990_MultilayerPerceptron is renamed to 8990_MLP.

5.1.1 AIRT algorithm metrics

Figure 19 shows the heatmaps of AIRT fitted probability distribution functions for the classification algorithms. We see that OLM and ConjunctiveRule are more stable comparatively. AIRT did not find any algorithm to be anomalous.

Table 4 gives AIRT metrics for the classification algorithms. Even though OLM has the highest algorithm consistency, it has the lowest difficulty limit. Therefore, OLM gives poor performances consistently. Thus, algorithm consistency by itself is not an indicator of a good algorithm. The RandomForest has the highest difficulty limit. Hence, the RandomForest can handle very difficult instances. Algorithms LMT, NaiveBayes, SMO_PolyKernel, AdaBoostM1_J48 and BayesNet also have high difficulty limits meaning that these algorithms can handle hard instances.

The RandomForest occupies the largest proportion in the latent trait (LTO) for $\epsilon = 0$ and the second largest for $\epsilon = 0.01$. Therefore, it is an excellent algorithm suited for a large number of diverse instances. Notably, LMT, the second best algorithm in terms of LTO for $\epsilon = 0$ surpasses the RandomForest and becomes the best algorithm for $\epsilon = 0.01$. This means, that even though it is not the topmost curve for most part of the latent trait, it is ϵ-close to the top curve mostly, and coupled with its own strengths on the latent trait it surpasses the RandomForest. Algorithm AdaBoostM1_J48 has a similar latent trait occupancy (LTO) as LMT when $\epsilon = 0$. Even though AdaBoostM1_J48's LTO increases when $\epsilon = 0.01$, it doesn't increase as much as LMT's LTO does. Curiously, REPTree and 8990_MLP have a similar proportion on the latent trait for both ϵ values. In contrast, algorithms such as J48, JRip and Bagging_REPTree, increase their LTO from 0 to values greater than 0.1 when ϵ increases from 0 to 0.01 – a bigger increase than REPTree and 8990_MLP undergo with the increase in ϵ. This observation suggests the two algorithms REPTree and 8990_MLP have unique strengths in the latent trait and not in other parts where more algorithms perform well.

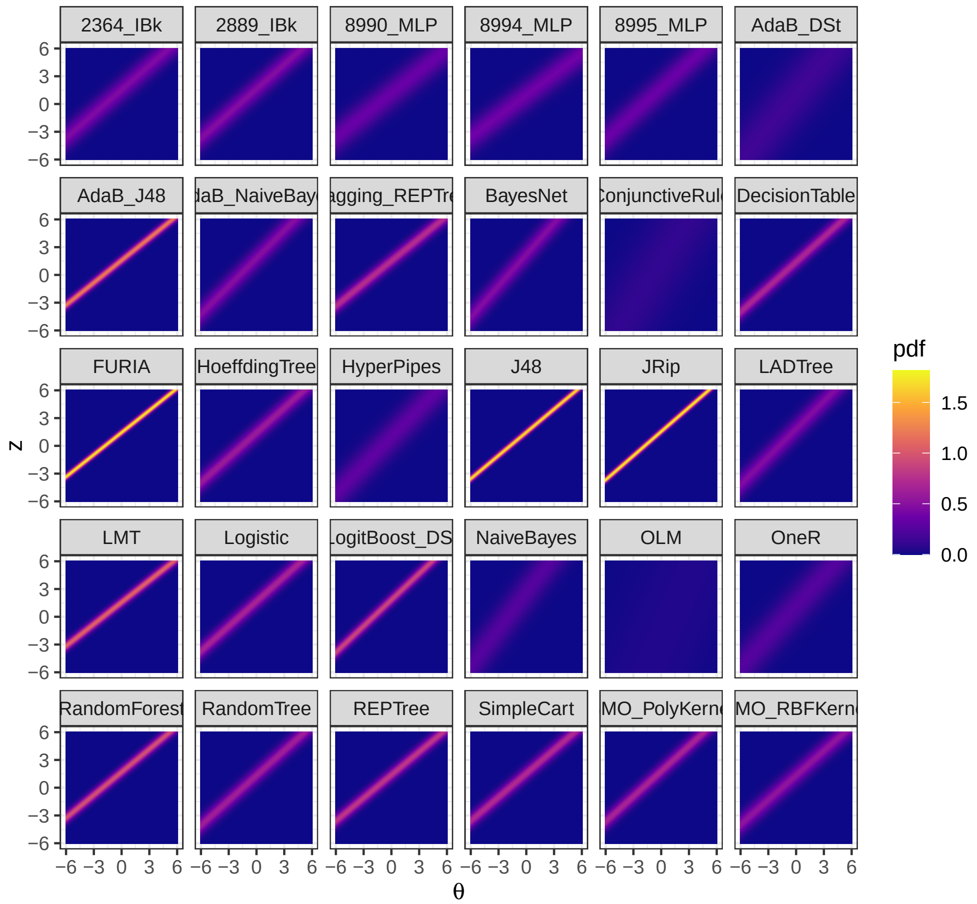


Figure 19: The heatmap of probability density functions for classification (OpenML-weka-2017) algorithms by fitting a continuous IRT model

5.1.2 Strengths and weaknesses of algorithms via AIRT

Figures 20 and 21 show the latent trait analysis for OpenML Weka classification algorithms. Figure 20 shows the performance of the algorithms with respect to problem difficulty and the resulting smoothing splines. The strengths and weaknesses of different algorithms are shown in Figure 21. The strengths and weaknesses are calculated for two values of ϵ, $\epsilon = 0$ and $\epsilon = 0.01$ as discussed in Section 4.2.

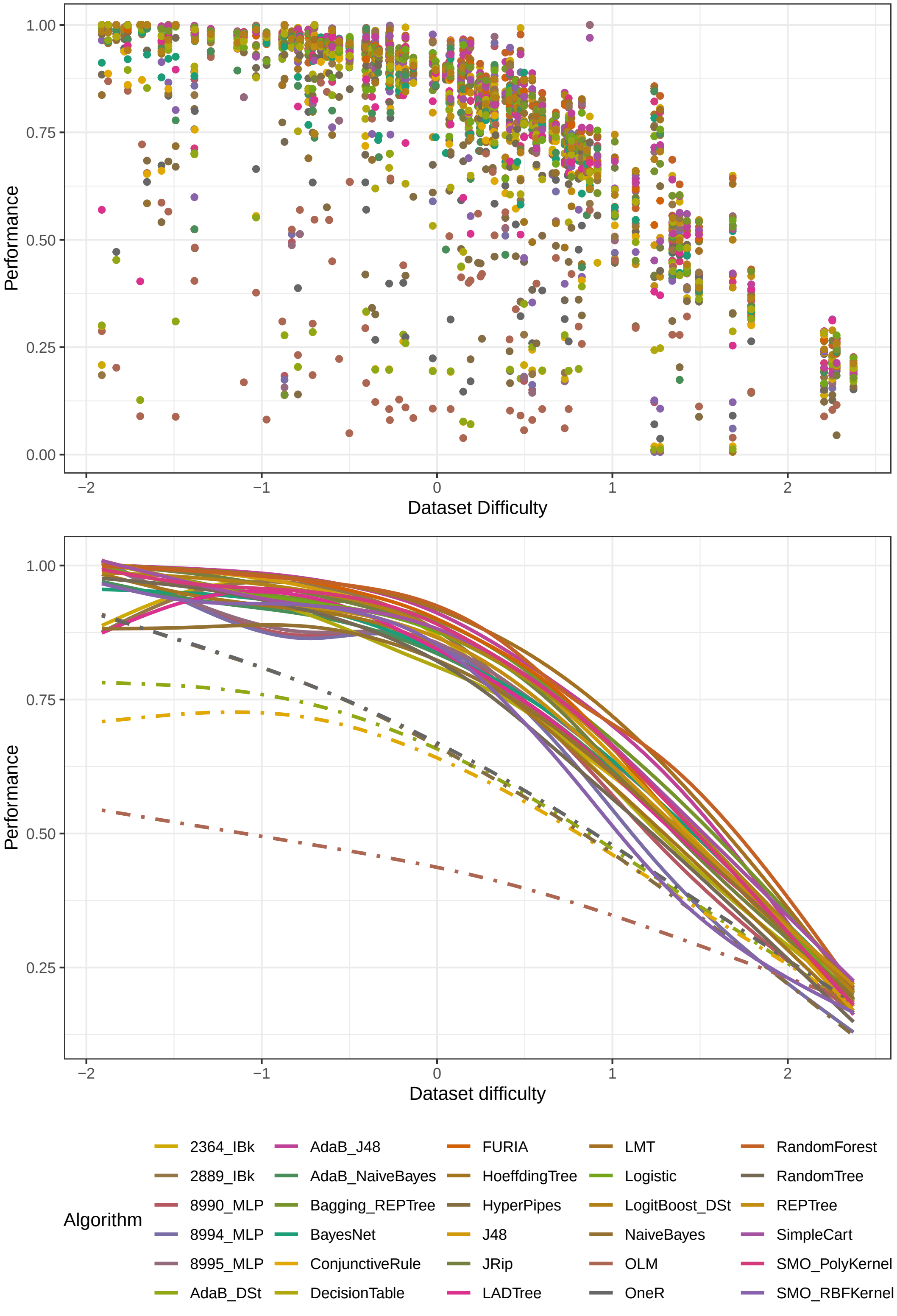


Figure 20: Algorithm performance with dataset/problem difficulty for classification algorithms. Top: Algorithm performance against dataset difficulty. Bottom: Latent trait curves for each algorithm with AdaB_DSt, ConjunctiveRule, HyperPipes, OLM and OneR in dashed lines.

Table 4: AIRT Metrics: algorithm consistency Score, Anomalousness indicator, Difficulty Limit and Latent Trait Occupancy (LTO) for classification algorithms.

Algorithm	Consistency	Difficulty Limit	Anomalousness	LTO ($\epsilon = 0$)	LTO ($\epsilon = 0.01$)
8990_MLP	1.401	1.427	FALSE	0.010	0.038
8994_MLP	1.349	1.271	FALSE	0.000	0.038
8995_MLP	1.201	1.842	FALSE	0.000	0.019
SMO_PolyKernel	0.655	1.950	FALSE	0.000	0.000
OneR	1.426	1.026	FALSE	0.000	0.000
J48	0.274	1.749	FALSE	0.000	0.162
2364_IBk	1.010	1.798	FALSE	0.000	0.000
REPTree	0.595	1.663	FALSE	0.029	0.076
RandomTree	0.709	1.457	FALSE	0.000	0.000
RandomForest	0.500	2.064	FALSE	0.410	0.790
LMT	0.467	1.994	FALSE	0.276	0.895
HoeffdingTree	0.757	1.553	FALSE	0.000	0.000
SMO_RBFKernel	0.842	1.508	FALSE	0.000	0.000
JRip	0.253	1.741	FALSE	0.000	0.124
2889_IBk	0.950	1.812	FALSE	0.000	0.000
HyperPipes	1.272	0.919	FALSE	0.000	0.000
NaiveBayes	1.173	1.968	FALSE	0.000	0.000
OLM	3.768	-1.176	FALSE	0.000	0.000
FURIA	0.281	1.806	FALSE	0.000	0.314
BayesNet	0.752	1.942	FALSE	0.000	0.000
ConjunctiveRule	2.473	0.845	FALSE	0.000	0.000
SimpleCart	0.643	1.819	FALSE	0.010	0.105
AdaBoostM1_NaiveBayes	0.819	1.750	FALSE	0.000	0.000
LADTree	0.852	1.793	FALSE	0.000	0.010
Logistic	0.669	1.824	FALSE	0.000	0.000
AdaBoostM1_DecisionStump	2.069	0.882	FALSE	0.000	0.000
AdaBoostM1_J48	0.408	1.947	FALSE	0.267	0.448
Bagging_REPTree	0.660	1.837	FALSE	0.000	0.105
DecisionTable	0.645	1.532	FALSE	0.000	0.067
LogitBoost_DecisionStump	0.473	1.927	FALSE	0.000	0.000

Of the 30 algorithms, 6 have strengths on the dataset difficulty spectrum when $\epsilon = 0$. These are 8990_MLP, AdaBoostM1_J48, LMT, RandomForest, REPTree and SimpleCart algorithms. In contrast 14 algorithms exhibit strengths when $\epsilon = 0.01$ showing the competitiveness of algorithms. The RandomForest displays strengths on a large region of the problem space followed by LMT when $\epsilon = 0.01$. We see that many algorithms have strengths for easy problems while not so many are strong for difficult problems. For the region when dataset difficulty is between 0.5 and 1, only LMT displays a strength. Similarly, when dataset difficulty is between 1.5 and 2, the RandomForest is the only algorithm that displays an advantage. In terms of weaknesses, OLM is weak for most of the

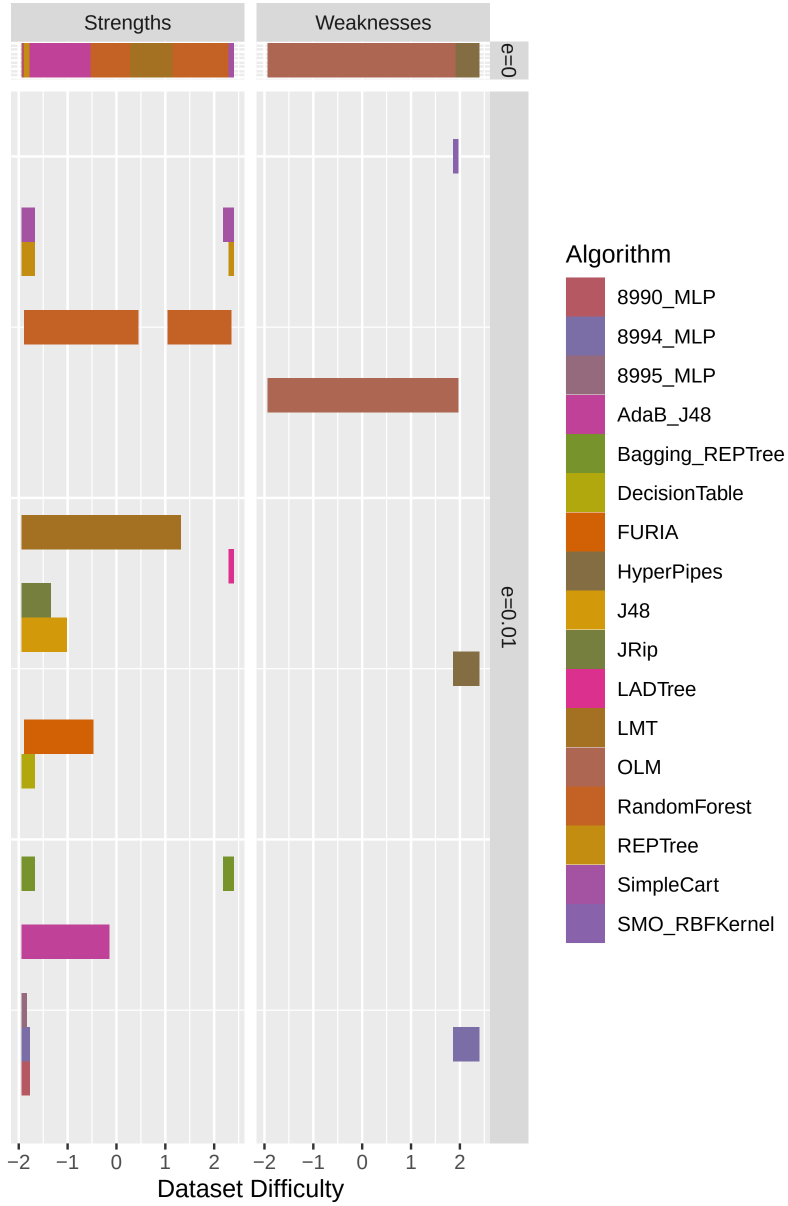


Figure 21: Strengths and weaknesses of OpenML Weka classification algorithms. The top bar shows the strengths and weaknesses for $\epsilon = 0$ and the bottom graph for $\epsilon = 0.01$.

problem space for both ϵ values. Hyperpipes are weak for more difficult problems for both ϵ values. The latent trait curves lying relatively below are shown in dashed lines so that they can be identified easier. These are AdaB_DSt, ConjunctiveRule, HyperPipes, OLM and OneR.

We can make some observations from Figure 21 and Table 4. The first is that the RandomForest and LMT cover almost all of the latent trait in the strengths diagram for $\epsilon = 0.01$. For $\epsilon = 0$, these two algorithms coupled with AdaB_J48 cover most of the strengths spectrum. Thus, these three algorithms, or even just RandomForest and LMT make a good combination in tackling diverse datasets. The second observation is that when increasing ϵ from 0 to 0.01, even though the number of algorithms increased from 6 to 14, most of them have strengths for very easy problems. Of the additional 8 algorithms, DecisionTable, 8994_MLP, 8995_MLP and LADTree have LTO < 0.1. Thus, we can disregard some of the algorithms with small LTO when $\epsilon = 0.01$. Considering the key algorithms, the main change from $\epsilon = 0$ to $\epsilon = 0.01$ is the increase in LTO for algorithm LMT.

5.1.3 AIRT model goodness metrics

Table 5 gives the model goodness results for classification algorithms. The MSE is less than 0.1 for all algorithms apart from OLM. Furthermore, the difference between predicted and actual effectiveness |AUPEC − AUAEC| is less than 0.1 for all algorithms apart from OLM, NaiveBayes and ConjunctiveRule. Figure 22 shows the effectiveness curves and the CDFs for this portfolio of algorithms. We see that most points on the AUAEC-AUPEC plane are close to the AUAEC = AUPEC line, which is shown by a dotted line. OLM is the exception. In general, the model has fitted the algorithm performances well.

5.1.4 Algorithm portfolio selection

We compare the airt portfolio with 2 additional algorithm portfolios:

1. *Shapley-portfolio*: a subset of algorithms selected using Shapley values (Fŕechette et al., 2016). Shapley values measure an algorithm's marginal contribution to the portfolio by using concepts from coalition game theory. For Shapley-portfolio we select algorithms with the top-n Shapley values.

2. *topset-portfolio*: a subset of algorithms having the best on-average performance at a per-instance level. The highest-ranked algorithm in the topset-portfolio gives the best performance for the most number of instances. For topset-portfolio we select the top-n best on-average algorithms

We construct Shapley, topset and airt portfolios with n algorithms and compare their performance for different values of n. As the evaluation metric we use the performance gap. Performance gap is computed using the best per-instance performance for each portfolio and the best per-instance performance using all the algorithms. We define the difference as the performance gap at a per-instance level. Let the best performance for instance i using the full set of algorithms be denoted by b_i. Let $\mathcal{A}_n, \mathcal{S}_n$ and $\mathcal{T}_n$ denote airt, Shapley and topset portfolios having n algorithms. Let $b_{\mathcal{A},i,n}$ denote the best performance for instance i using the airt portfolio with n algorithms. Similarly, let $b_{\mathcal{S},i,n}$ and $b_{\mathcal{T},i,n}$ denote the best performance for instance i using Shapley and topset portfolios with n algorithms. Then we define the performance gap for instance i for each portfolio as

$$\text{Perf. gap}_{\mathcal{A},i,n} = b_i - b_{\mathcal{A},i,n}\,, \quad \text{Perf. gap}_{\mathcal{S},i,n} = b_i - b_{\mathcal{S},i,n} \quad \text{and} \quad \text{Perf. gap}_{\mathcal{T},i,n} = b_i - b_{\mathcal{T},i,n}\,.$$

For each algorithm portfolio and n we get an $N \times 1$ vector of performance gap values. We compute the mean performance gap for each n. For each algorithm scenario we use 10-fold cross validation and report the average cross validated performance gap for Shapley, topset and airt portfolios. Additionally, we compute the standard errors using different folds. We note that *Perf. gap* is the

Table 5: MSE, AUCDF, Area Under Actual Effectiveness Curve (AUAEC) and Predicted Effectiveness Curves (AUPEC) and |AUAEC - AUPEC| for classification algorithms.

Algorithm	MSE	AUCDF	AUAEC	AUPEC	|AUAEC - AUPEC|
8990_MLP	0.032	0.863	0.758	0.730	0.028
8994_MLP	0.036	0.844	0.747	0.700	0.047
8995_MLP	0.029	0.899	0.776	0.809	0.033
SMO_PolyKernel	0.007	0.940	0.814	0.820	0.006
OneR	0.051	0.840	0.637	0.712	0.075
J48	0.004	0.944	0.810	0.782	0.028
2364_IBk	0.022	0.910	0.785	0.795	0.010
REPTree	0.010	0.932	0.794	0.769	0.025
RandomTree	0.007	0.939	0.754	0.754	0.000
RandomForest	0.006	0.938	0.845	0.816	0.029
LMT	0.007	0.928	0.848	0.800	0.048
HoeffdingTree	0.011	0.921	0.768	0.767	0.001
SMO_RBFKernel	0.017	0.899	0.750	0.759	0.009
JRip	0.003	0.950	0.805	0.787	0.018
2889_IBk	0.020	0.920	0.789	0.800	0.011
HyperPipes	0.040	0.846	0.629	0.683	0.054
NaiveBayes	0.029	0.868	0.751	0.881	0.130
OLM	0.164	0.681	0.411	0.171	0.240
FURIA	0.006	0.929	0.824	0.780	0.044
BayesNet	0.011	0.920	0.782	0.855	0.073
ConjunctiveRule	0.093	0.784	0.588	0.719	0.131
SimpleCart	0.009	0.943	0.811	0.794	0.017
AdaBoostM1_NaiveBayes	0.013	0.928	0.771	0.813	0.042
LADTree	0.012	0.938	0.774	0.818	0.044
Logistic	0.005	0.942	0.805	0.802	0.003
AdaBoostM!_DSt	0.082	0.794	0.611	0.698	0.087
AdaBoostM1_J48	0.006	0.934	0.837	0.798	0.039
Bagging_REPTree	0.011	0.929	0.820	0.787	0.033
DecisionTable	0.009	0.931	0.761	0.764	0.003
LogitBoost_DSt	0.004	0.956	0.812	0.824	0.012

same as *misclassification penalty* discussed in Bischl et al. (2016). However, we have used the term *Perf. gap* because we think it is more intuitive and applicable to non-classification scenarios.

Figure 23 shows the mean performance gap of the 3 portfolios using 10-fold cross validation for OpenML Weka algorithms for different values of ϵ. A lower gap is preferred as it indicates the portfolio has better algorithms. The vertical lines at each point show the standard errors. We see that airt generally has lower performance gaps. The number of algorithms in the airt portfolio changes with ϵ. For each ϵ, as the limiting number of algorithms (the maximum x value) we select the minimum number of algorithms from airt, Shapley and topset. For $\epsilon = 0$ airt selects 6 algorithms,

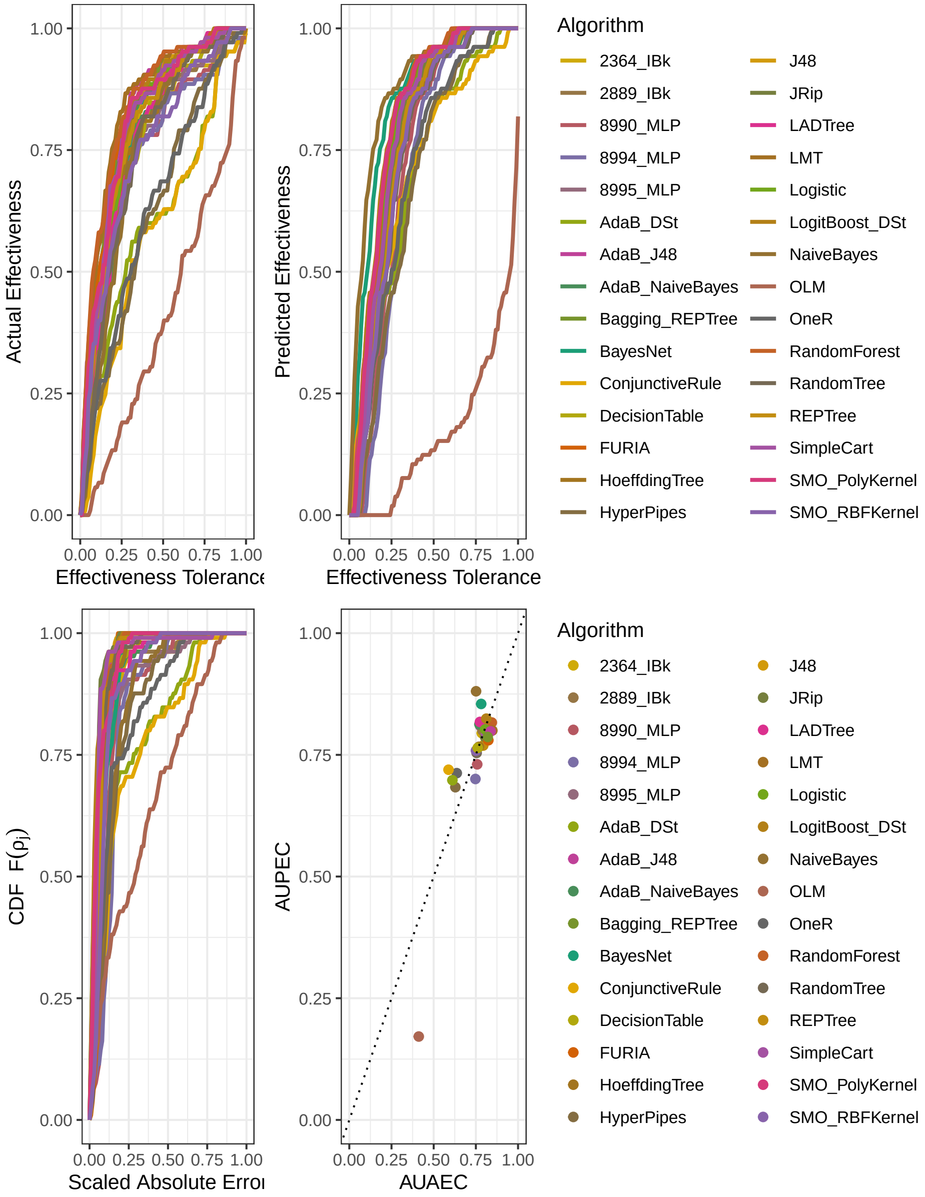


Figure 22: Model goodness metrics for classification algorithms: the actual and predicted effectiveness curves on the top row and the CDFs $F(\rho_j)$ and (AUAEC, AUPEC) on the bottom row.

which decides the limiting number of algorithms. For other ϵ values Shapley decides the limiting number of algorithms in this example. For each fold, different algorithms may get selected by different portfolio selection methods. Thus, for $n = 14$ standard errors are not computed because 14 algorithms are selected only in 1 fold.

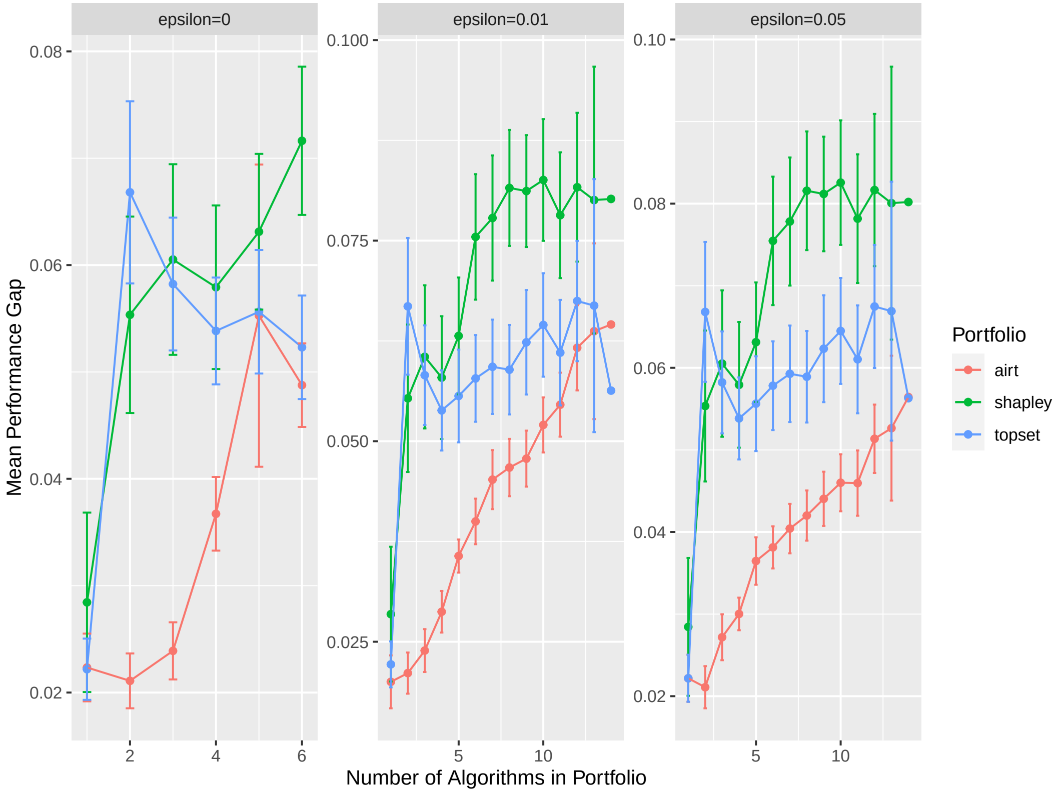


Figure 23: Performance analysis of Shapley, topset and airt portfolios for different ϵ values. The mean cross-validated performance gap is shown with standard errors denoted by vertical lines.

Table 6: Additional ASlib case studies. Mean Performance Gap (MPG) of a portfolio of 5 algorithms is reported using 10-fold CV with the best in bold.

Scenario	Measurement	Num. Obs.	Num. Algorithms	airt MPG	Shapley MPG	topset MPG
OPENML_WEKA	accuracy	105	31	**0.0553**	0.0631	0.0556
ASP_POTASSCO	runtime	1294	11	78.0	92.7	**77.8**
CSP_MINIZINC_2016	par10	100	21	**1962**	2371	2026
GRAPHS_2015	runtime	5725	8	**1689346**	6127229	6763210
MAXSAT_PMS_2016	par10	601	20	**1019**	1469	1305
PROTEUS_2014	runtime	4021	23	**293**	648	1125
SAT11_INDU	runtime	300	19	882	**826**	855
SAT12_ALL	runtime	1614	32	**456**	523	683
SAT18_EXP_ALGO	runtime	353	38	**1677**	1823	1822
BNSL_2016	runtime	1179	9	**1210**	1448	2030

5.2 Additional case studies

We conduct shorter analyses for 9 additional ASlib scenarios, which are given in Appendix A. We explore the latent trait curves, strengths and weaknesses of algorithms for $\epsilon \in \{0, 0.05\}$ and compare different algorithm portfolios. As each scenario other than OPENML-Weka has runtimes or par10 values as the evaluation metric, we transform these values by multiplying with -1 and scaling to the interval $[0, 1]$ with 1 denoting good performance and 0 denoting poor performance. Some summary statistics of these analyses are given in Table 6. As it is difficult to encapsulate the strengths and weaknesses or the latent trait curves by a single numeric value, we give the mean performance gap of the portfolios with 5 algorithms in Table 6. Figures of the mean performance gap for different number of algorithms with standard errors and other details are given in the Appendix.

For most scenarios airt performs well. Even though airt does not perform well for SAT11_INDU, from the MPG curves in the Appendix we see that the standard errors of the different portfolios overlap. We also see that the latent trait curves for SAT11_INDU are all bundled up together. This tells us that SAT11_INDU algorithms are similar in performance. From other scenarios, we notice that airt is better at identifying a good portfolio of algorithms when algorithms are diverse, i.e., when the latent trait curves display high variability. The construction of the latent trait θ involves IRT discrimination and difficulty parameters as well as the actual performance. Therefore, selecting algorithms based on fitting splines to θ takes into account this underlying hidden quantity uncovered by IRT that denotes the dataset difficulty spectrum. This allows us to select a good portfolio of algorithms when a diverse set of algorithms are present. This is another use of AIRT in addition to its exploratory aspect.

6. Conclusions

Beyond standard statistical analysis, which often hides useful insights, there are not many techniques that can be used to rigorously evaluate a portfolio of algorithms and identify their strengths and weaknesses. One such technique is the instance space analysis methodology which can be used to visualize the strengths and weaknesses of algorithms. As the instance space incorporates both the algorithms and the test instances, computing features of test instances is an essential step to constructing an instance space. Devising suitable features of test instances that capture their intrinsic difficulties for algorithms is a significant challenge that can limit the applicability of the method. In this paper we have taken a different approach to achieve the same goal that avoids the need to devise instance features. We have presented AIRT, an IRT based algorithm evaluation method, which evaluates algorithms using only performance results. We demonstrated its usefulness on a diverse set of algorithm portfolios arising from a wide variety of problem domains. The scenarios used are taken from the ASlib repository containing algorithm implementations with given parameter and hyperparameter settings. We have not explored different parameter settings in this study and this is a limitation. Each parameter setting would give rise to a different algorithm implementation that would result in a different algorithm curve. Thus, by considering a single algorithm with different parameter settings, AIRT has potential to select parameter settings that are advantageous for easy or difficult problems.

Recasting the IRT framework as an inverted model, AIRT focuses on evaluating algorithm attributes such as consistency, anomalousness and difficulty limit thereby helping to broaden the understanding of algorithm behaviors and their dependence on test instances. AIRT can be used to visualize the strengths and weaknesses of algorithms in different parts the problem space. Using algorithms with strengths we construct an algorithm portfolio and show that it has a low performance gap compared to other portfolios. In addition, IRT model goodness measures can be derived, showing the level of trustworthiness of the underlying IRT model. Due to the fact that AIRT extends the IRT framework, it also has the desirable mathematical and optimality properties inherited from the embedded maximum likelihood estimation techniques. Furthermore, the explainable nature of IRT parameters gets translated to the algorithm evaluation domain.

As future research avenues we plan to consider the role of AIRT in parameter selection and alternative remappings of the IRT framework to increase understanding of the strengths and weaknesses of dataset repositories, thereby providing means to select an unbiased yet diverse collection of datasets, drawing deeper insights into their abilities to support meaningful conclusions about algorithm strengths and weaknesses.

Acknowledgments

Funding was provided by the Australian Research Council through the Australian Laureate Fellowship FL140100012, and the ARC Training Centre in Optimisation Technologies, Integrated Methodologies and Applications (OPTIMA) under grant IC200100009. The authors would like to thank Prof Rob J. Hyndman for his suggestion of the name AIRT for our method.

Supplementary Material

The algorithm performance datasets used in this paper are found at `https://github.com/coseal/aslib_data` and the programming scripts using AIRT are available at `https://github.com/sevvandi/airt-scripts`.

References

Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. ASlib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016. ISSN 00043702. doi: 10.1016/j.artint.2016.04.003.

Giuseppe Casalicchio, Jakob Bossek, Michel Lang, Dominik Kirchhoff, Pascal Kerschke, Benjamin Hofner, Heidi Seibold, Joaquin Vanschoren, and Bernd Bischl. Openml: An r package to connect to the machine learning platform openml. *Computational Statistics*, 34(3):977–991, 2019.

R. Chalmers. mirt: A Multidimensional Item Response Theory Package for the R Environment. *Journal of Statistical Software, Articles*, 48(6):1–29, 2012. ISSN 1548-7660. doi: 10.18637/jss.v048.i06.

Yu Chen, Ricardo BC Prudêncio, Tom Diethe, Peter Flach, et al. β^3-IRT: A New Item Response Model and its Applications. *arXiv preprint, arXiv:1903.04016*, 2019.

Andrew Cooper and Konstantinos Vassilis Petrides. A psychometric analysis of the Trait Emotional Intelligence Questionnaire–Short Form (TEIQue–SF) using item response theory. *Journal of Personality Assessment*, 92(5):449–457, 2010.

A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011. ISSN 22106502. doi: 10.1016/j.swevo.2011.02.001.

Susan E Embretson and Steven P Reise. *Item Response Theory*. Psychology Press, 2013.

Alexandre Fréchette, Lars Kotthoff, Tomasz Michalak, Talal Rahwan, Holger H. Hoos, and Kevin Leyton-Brown. Using the shapley value to analyze algorithm portfolios. In *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pages 3397–3403, 2016. ISBN 9781577357605.

Bernadette Gray-Little, Valerie S.L. Williams, and Timothy D. Hancock. An item response theory analysis of the Rosenberg self-esteem scale. *Personality and Social Psychology Bulletin*, 23(5): 443–451, 1997. ISSN 01461672. doi: 10.1177/0146167297235001.

M Hall, E Frank, G Holmes, B Pfahringer, P Reutemann, and I H Witten. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009. ISSN 1931-0145.

Nicholas G Hall and Marc E Posner. The generation of experimental data for computational testing in optimization. In *Experimental methods for the analysis of optimization algorithms*, pages 73–101. Springer, 2010.

Ronald K Hambleton and Hariharan Swaminathan. *Item Response Theory: Principles and Applications*. Springer Science & Business Media, 2013.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

John N Hooker. Needed: An empirical science of algorithms. *Operations research*, 42(2):201–212, 1994.

John N Hooker. Testing heuristics: We have it all wrong. *Journal of heuristics*, 1(1):33–42, 1995.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

Sevvandi Kandanaarachchi. *airt: Evaluation of Algorithm Collections Using Item Response Theory*, 2020. URL `https://cran.r-project.org/web/packages/airt/index.html`. R package version 0.1.0.

Sevvandi Kandanaarachchi. Unsupervised anomaly detection ensembles using item response theory. *Information Sciences*, 587:142–163, 2022. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2021.12.042. URL `https://www.sciencedirect.com/science/article/pii/S0020025521012639`.

Sevvandi Kandanaarachchi, Mario A Muñoz, Rob J Hyndman, and Kate Smith-Miles. On normalization and algorithm selection for unsupervised outlier detection. *Data Mining and Knowledge Discovery*, 34:309—354, 2019. doi: https://doi.org/10.1007/s10618-019-00661-z.

Y. Kang, R.J. Hyndman, and K. Smith-Miles. Visualising forecasting algorithm performance using time series instance spaces. *International Journal of Forecasting*, 33(2):345–358, 2017. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2016.09.004.

Markelle Kelly, Aakriti Kumar, Padhraic Smyth, and Mark Steyvers. Capturing Humans' Mental Models of AI: An Item Response Theory Approach. In *FAccT '23: 2023 ACM Conference on Fairness, Accountability, and Transparency, Chicago, IL, USA, June 2023*, pages 1723–1734, 2023. doi: 10.1145/3593013.3594111.

Christiane Lemke, Marcin Budka, and Bogdan Gabrys. Metalearning: a survey of trends and technologies. *Artificial Intelligence Review*, 44(1):117–130, 2015. ISSN 15737462. doi: 10.1007/s10462-013-9406-y.

David Lewis. Causal Explanation. In *Philosophical Papers Vol. Ii*, pages 214–240. Oxford University Press, 1986.

Marius Lindauer, Jan N. van Rijn, and Lars Kotthoff. Open Algorithm Selection Challenge 2017: Setup and Scenarios. In *Proceedings of the Open Algorithm Selection Challenge*, volume 79 of *Proceedings of Machine Learning Research*, pages 1–7. PMLR, 11–12 Sep 2017.

Frederic M Lord. *Applications of Item Response Theory to practical testing problems*. Routledge, 1980.

Nuria Marcia and Ester Bernad´ o Mansilla. Towards UCI+: a mindful repository design. *Information Sciences*, 261(10):237–262, 2014.

Fernando Martínez-Plumed, Ricardo BC Prudêncio, Adolfo Martínez-Usó, and José Hernández-Orallo. Item Response Theory in AI: Analysing machine learning classifiers at the instance level. *Artificial Intelligence*, 271:18–42, 2019.

Catherine C McGeoch. Toward an experimental method for algorithm simulation. *INFORMS Journal on Computing*, 8(1):1–15, 1996.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2018.07.007. URL `https://www.sciencedirect.com/science/article/pii/S0004370218305988`.

M.A. Muñoz and K.A. Smith-Miles. Performance analysis of continuous black-box optimization algorithms via footprints in instance space. *Evol. Comput.*, 25(4):529–554, 2017. doi: 10.1162/EVCO_a_00194.

Mario A. Muñoz, Laura Villanova, Davaatseren Baatar, and Kate Smith-Miles. Instance spaces for machine learning classification. *Machine Learning*, 107(1):109–147, 2018.

Oxford English Dictionary, June 2016. URL `https://www.oed.com/view/Entry/66604`. Accessed on 2023-07-17.

Mark D Reckase. Multidimensional item response theory models. In *Multidimensional item response theory*, pages 79–112. Springer, 2009.

John R. Rice et al. The algorithm selection problem. *Advances in computers*, 15(65-118):5, 1976.

Dimitris Rizopoulos. ltm: An R Package for Latent Variable Modeling and Item Response Analysis. *Journal of Statistical Software, Articles*, 17(5):1–25, 2006. ISSN 1548-7660. doi: 10.18637/jss.v017.i05.

Fumiko Samejima. Estimation of latent ability using a response pattern of graded scores. *Psychometrika monograph supplement*, 34:1–97, 1969.

Fumiko Samejima. Homogeneous case of the continuous response model. *Psychometrika*, 38(2): 203–219, 1973. ISSN 00333123.

Fumiko Samejima. Normal ogive model on the continuous response level in the multidimensional latent space. *Psychometrika*, 39(1):111–121, 1974. ISSN 00333123. doi: 10.1007/BF02291580.

Galit Shmueli. To explain or to predict? *Statistical Science*, 25(3):289–310, 2010. ISSN 08834237. doi: 10.1214/10-STS330.

Kojiro Shojima. A noniterative item parameter solution in each EM cycle of the continuous response model. *Educational technology research*, 28(1):11–22, 2005. ISSN 0387-7434.

Kate Smith-Miles and Simon Bowly. Generating new test instances by evolving in instance space. *Computers & Operations Research*, 63:102–113, 2015.

Kate Smith-Miles and Thomas T Tan. Measuring algorithm footprints in instance space. In *2012 IEEE Congress on Evolutionary Computation*, pages 3446–3453. IEEE, 2012.

Kate Smith-Miles, Davaatseren Baatar, Brendan Wreford, and Rhyd Lewis. Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, 45:12–24, 2014.

Wim J van der Linden and Ronald K Hambleton. *Handbook of modern Item Response Theory*. Springer Science & Business Media, 2013.

J N van Rijn. *Massively Collaborative Machine Learning*. PhD thesis, 2016. URL `https://openaccess.leidenuniv.nl/handle/1887/44814`.

Ricardo Vilalta, Christophe Giraud-Carrier, and Pavel Brazdil. Meta-learning-concepts and techniques. In *Data mining and knowledge discovery handbook*, pages 717–731. Springer, 2009.

Tianyou Wang and Lingjia Zeng. Item parameter estimation for a continuous response model using an EM algorithm. *Applied Psychological Measurement*, 22(4):333–344, 1998. ISSN 01466216.

Lin Xu, Frank Hutter, Jonathan Shen, and HH Hoos. SATzilla2012: Improved Algorithm Selection Based on Cost-sensitive Classification Models. In *Proceedings of SAT*, 2012.

Wendy M. Yen. the Choice of Scale for Educational Measurement: an Irt Perspective. *Journal of Educational Measurement*, 23(4):299–325, 1986. ISSN 17453984. doi: 10.1111/j.1745-3984.1986.tb00252.x.

Yvonnick Noel and Bruno Dauvier. A Beta Item Response Model for Continuous Bounded Responses. *Applied Psychological Measurement*, 31(1):47–73, 2007.

Appendix A. ASlib scenarios

In this section we explore 9 ASlib scenarios: ASP-POTASSCO, CSP-MiniZinc-Time-2016, GRAPHS-2015, MAXSAT-PMS-2016, PROTEUS-2014, SAT11-INDU, SAT12-ALL, BNSL-2016 and SAT18-EXP-ALGO. For each scenario we fit an AIRT model and conduct a smaller analysis compared to the OpenML-Weka example in Section 5.1. Using the fitted model, we plot the latent trait curves. Then we compute the strengths and weaknesses of algorithms on the dataset difficulty spectrum for $\epsilon = 0$ and $\epsilon = 0.05$. By visualizing this spectrum, we see which algorithms have strengths for easy problems and which ones are better suited for difficult problems. Similarly, we see their weaknesses as well. Using 10-fold cross validation, we evaluate airt, topset and Shapley algorithm portfolios and examine the mean performance gap as explained previously.

A.0.1 ASP_POTASSCO

Figure 24 shows the analysis for ASP_POTASSCO scenario. Algorithm *clasp/2.1.3/h3-n1* is the weakest in the portfolio as we can see from the strengths and weaknesses figure and the latent trait curves. Algorithm *clasp/2.1.3/h1-n1* is better suited for difficult problems as seen by the hump in the latent trait curve around $\delta \approx 1$. Many algorithms perform well for very easy problems as seen by the leftmost part of the strengths in the problem difficulty spectrum. A point of interest about these curves is that some curves, including that of *clasp/2.1.3/h1-n1*, have a turning point around $\delta \approx 0.75$ followed by a positive slope, signifying an improvement in performance, culminating at $\delta \approx 1.25$ before decreasing again. This shows locally anomalous behavior for δ in that region for certain algorithms. The cross-validated mean performance gap of different algorithm portfolios show that for $n \in \{1, \ldots, 5\}$ airt is similar to either Shapley or topset, but for $n \in \{6, \ldots, 9\}$ airt has a lower mean performance gap. However, the standard errors show that the differences are not significant.

A.0.2 CSP_MiniZinc_2016

Figure 25 shows the analysis for CSP_MiniZinc_2016. The latent trait curves are spread out well and thus show high variability. This has resulted in a sparse set of strengths and weaknesses. Even though there are many algorithms, only a few exhibit strengths and similarly only a few have weaknesses at other places apart from the rightmost end, which has the most difficult problems. As seen from the latent trait curves and the strengths and weaknesses figure, algorithm *LCG-Glucose-UC-free* shows continued strength for difficult and semi-difficult problems. Algorithm *MZN/Gurobi-free* is better suited for easy and very difficult problems. The weakest algorithm is *Picat-CP-fd*, which is weak for easy and semi-difficult problems. While many algorithms are good for easy problems, both *LCG-Glucose-UC-free* and *LCG-Glucose-free* displays strengths for a large region of the problem space for $\epsilon = 0.05$. The cross-validated mean performance gap graphs show that airt and topset behave similarly, while Shapley has higher mean performance gaps initially but converges with airt and topset for higher n.

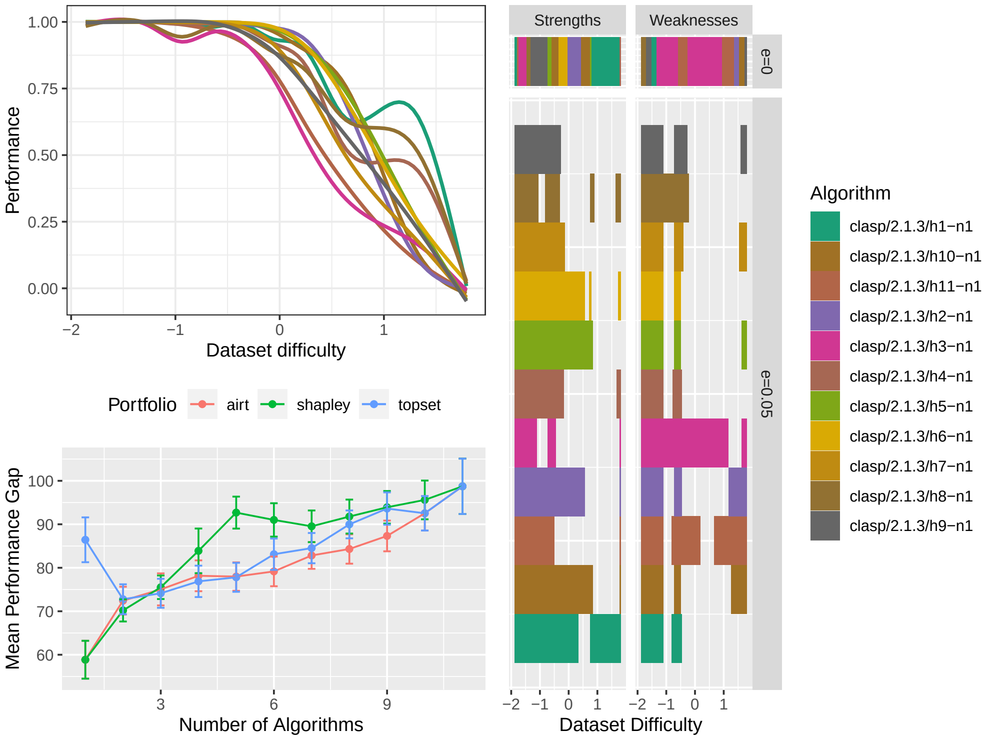


Figure 24: Latent trait curves, strengths and weaknesses and 10-fold CV portfolio comparison for ASP_POTASSCO scenario.

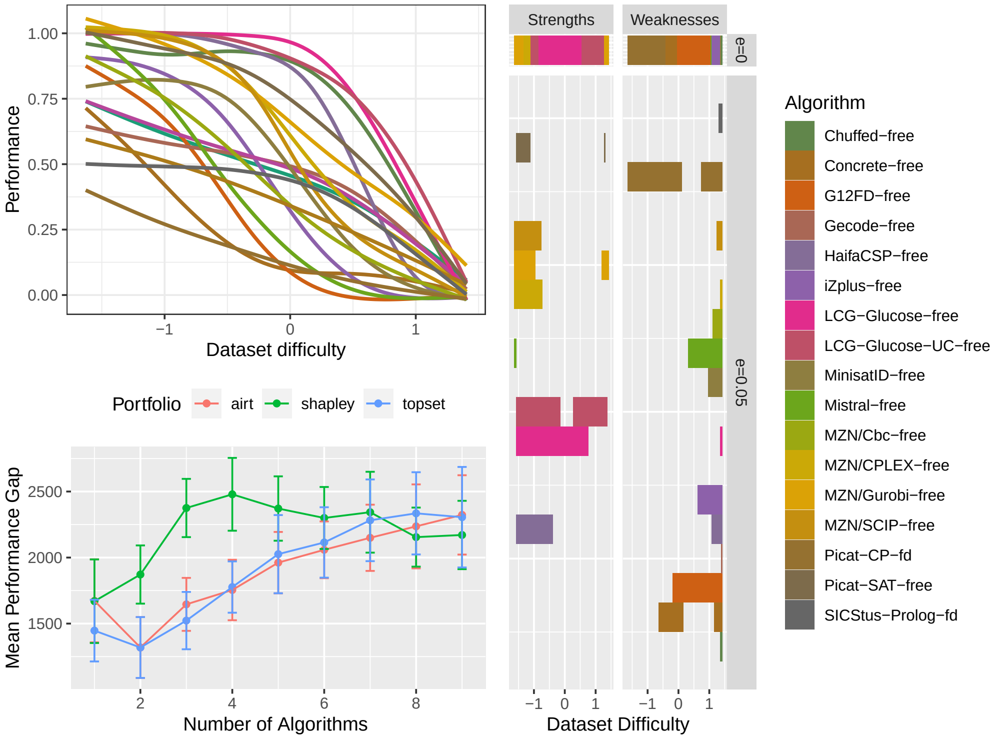


Figure 25: Latent trait curves, strengths and weaknesses and 10-fold CV portfolio comparison for CSP_MiniZinc scenario.

A.0.3 Graphs_2015

Figure 26 shows the analysis for Graphs_2015. From the latent trait curves and the strengths and weaknesses figure we see that *glasgow2* and *glasgow3* are suited for a large part of the problem space. *supplemantallad* is good for easy and very difficult problems and many algorithms have strengths for easy problems. The weakest algorithm is *vf2* as seen from the weaknesses spectrum. For dataset difficulty $\delta \lessapprox 0.5$, all algorithms apart from *vf2* perform well. However, after that point, the algorithms diverge in their performance as seen from the curves. The cross-validated mean performance gap of different portfolios show that airt has the smallest performance gap for most n.

A.0.4 MAXSAT-PMS-2016

Figure 27 shows the analysis for MaxSAT-PMS-2016 scenario. Immediately we see variety in the latent trait curves. Some curves have low performance values for most part of the space, which is different from the other scenarios we examined so far. Some curves have varying behavior with curved sections. In the strengths diagram, we see many algorithms having strengths for easier problems. Of the algorithms, 15 have strengths for dataset difficulty $\delta \leq 0$ when $\epsilon = 0.05$. These are the easy problems. For the easy problems, any of these algorithms would give good performances. Only 8 algorithms have strengths for $0 \leq \delta \leq 1$ and of these only 5 have strengths for $\delta > 1$. *LMHS-2016* and *maxhs-b* are better suited for harder problems. In the weaknesses space we see that *CCLS2akms* and *CCEHC2akms* are very weak algorithms. The cross-validated mean performance gap shows that airt performs better compared to the other two portfolios. The topset portfolio has a sudden jump at $n = 6$, possibly due to including a volatile algorithm, which gets mitigated with subsequent algorithm additions to the portfolio.

A.0.5 PROTEUS-2014

The latent curves of PROTUES-2014, shown in Figure 28 have many wiggles. Four curves achieve local minima at dataset difficulty $\delta \approx -0.2$. After that point, their performance increase for some part of the dataset difficulty spectrum, i.e., as the dataset difficulty increases, the performance of these algorithms get better. Thus, these algorithms are locally anomalous. They are not anomalous throughout the spectrum, but they have regions of locally anomalous behaviour. Algorithms *claspcnf_support, claspcnf_direct* and *claspcnf_directorder* display strengths for a large part of the problem space including difficult problems. Algorithm *gecode* is the weakest algorithm as seen by the latent trait curves and the strengths and weaknesses diagram. The cross-validated mean performance gap curves show that airt performs better than the other two portfolios. The standard errors for both topset and airt are very low making them not clearly visible in the diagram.

A.0.6 SAT11-INDU

Figure 29 shows the analysis of SAT11-INDU. We see that most algorithms have similar-shaped latent trait curves. We do not know if the algorithms were preselected, which might account for this behaviour. The similarity of the curves implies some similarity of performance between the algorithms. In the strengths diagram many algorithms have strengths for easy and semi-difficult problems. In the weaknesses diagram, we see a curious occurrence: many algorithms display weaknesses in the middle of the spectrum as well as on the difficult end of the spectrum. This is because the curves are packed together for most part of the problem space. Algorithm *glucose_2* occupies

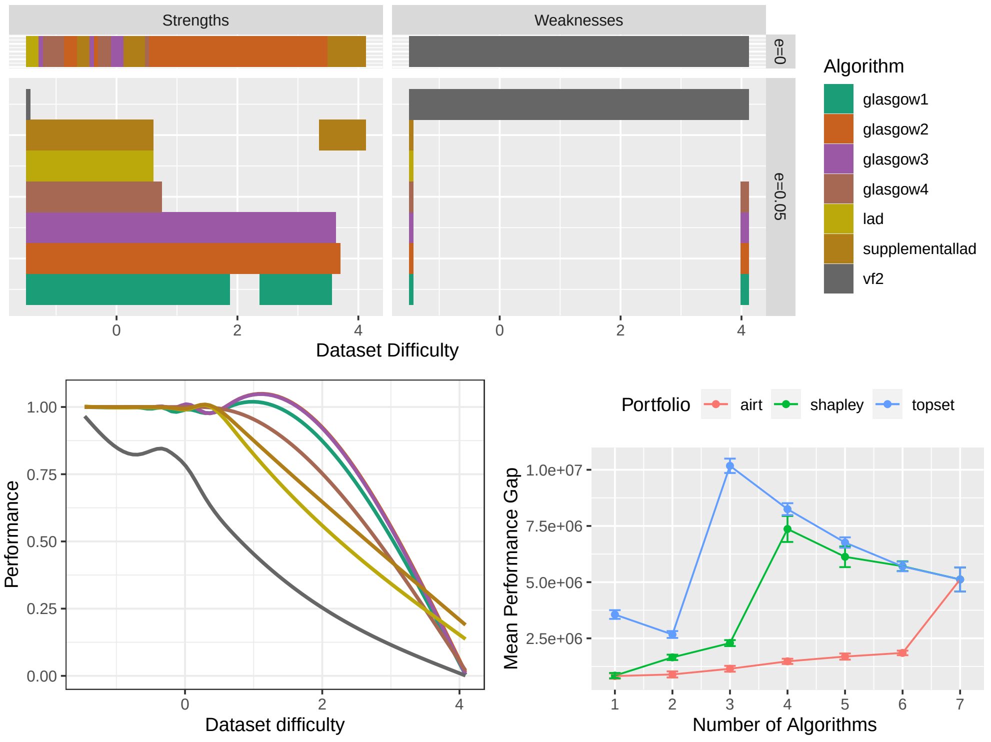


Figure 26: Strengths and weaknesses, latent trait curves and 10-fold CV portfolio comparison for Graphs_2015 scenario.

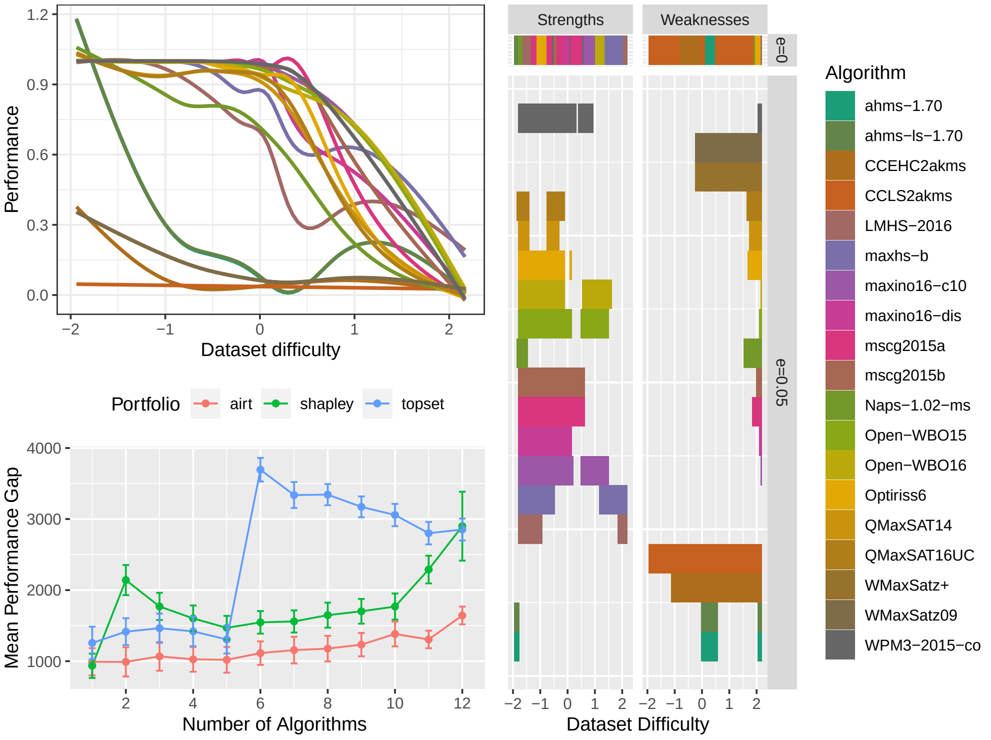


Figure 27: Strengths and weaknesses, latent trait curves and 10-fold CV portfolio comparison for MAXSAT_PMS_2016 scenario.

the highest proportion of the latent trait. From the strengths and weaknesses figure we see that *QuteRSat_2011-05-12_fixed* is strong for difficult problems. Notably *minisathackcontrasat_2011-03-02* is weak for easy problems. The cross-validated performance gap curves show that Shapley performs better than the others, but the standard errors of the 3 portfolios overlap for most values of n.

A.0.7 SAT12-ALL

SAT12-ALL scenario contains SATzilla 2012 competition (Xu et al., 2012) results on algorithm performance. Figure 30 shows the latent trait curves, strengths and weaknesses and performance comparison of different portfolios. The curves have diverse characteristics: some curves have an initial downward trend showing that they are weak for most part of the space but later trend upward indicating that they perform better for more difficult test instances. Another set of curves give good performances for easy problems with $\delta \lessapprox -1$ and decrease in performance after that. Algorithms *mphaseSATm* and *mphaseSAT* are strong for a large part of the problem space including difficult instances. Algorithms *spear-sw* and *eagleup* are weak for most parts of the space. The cross-validated mean performance gap curves show that airt has a lower gap compared to the other 2 portfolios.

A.0.8 BNSL-2016

Figure 31 shows the analysis for BNSL-2016 scenario. Algorithms ilp-141 and ilp-141-nc have similar latent trait curves. Similarly, ilp-162 and ilp-162-nc are also similar. Furthermore, astar-ec and astar-ed3 have similar curves. Lastly, cpbayes and astar-comp have somewhat similar curves. Algorithms *cpbayes* and *astar-comp* display strengths for easy and very difficult problems while *astar-ec* and *astar-ed3* are weak for most of the problem space. Algorithms ilp-141, ilp-141-nc, ilp-162 and ilp-162-nc have strengths for a large part of the problem space. Algorithm portfolio comparison shows that airt achieves good performance.

A.0.9 SAT18-EXP-ALGO

Figure 32 shows the analysis for SAT18_EXP_ALGO scenario. The latent trait curves are somewhat similar, but not too similar as in SAT11_INDU. The algorithm *YalSAT*, depicted by a gray shade, is weak for easier instances and strong for difficult instances. Hence is comes up in both strengths and weakness diagrams. Remarkably, the latent trait curve appears at the bottom on the left hand side and bends and ends up at the top at the right-most side. Another upward bend is observed at $\delta \approx -0.5$ by Maple_CM_Dist algorithm showing a unique strength of this algorithm. The airt portfolio achieves good performance for this scenario.

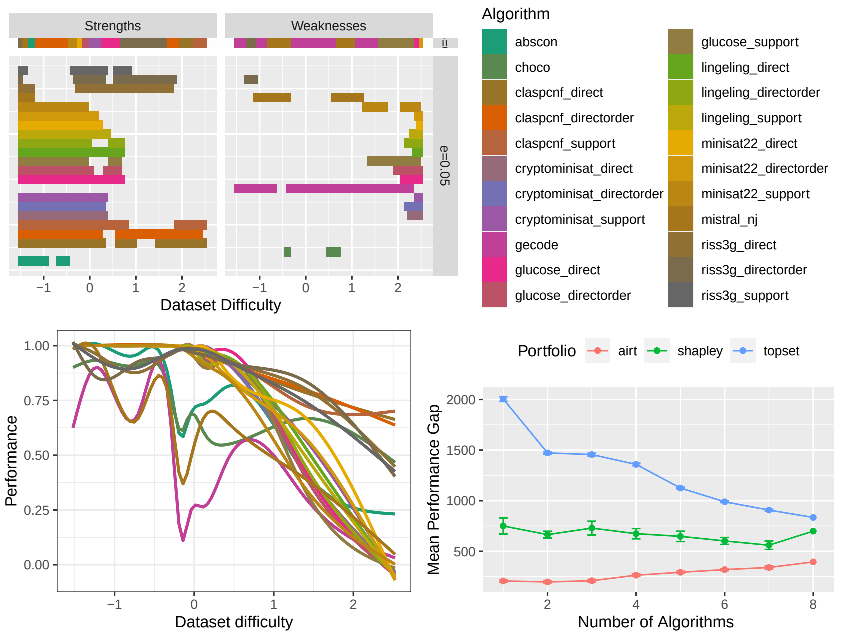


Figure 28: Strengths and weaknesses, latent trait curves and 10-fold CV portfolio comparison for PROTEUS_2014 scenario.

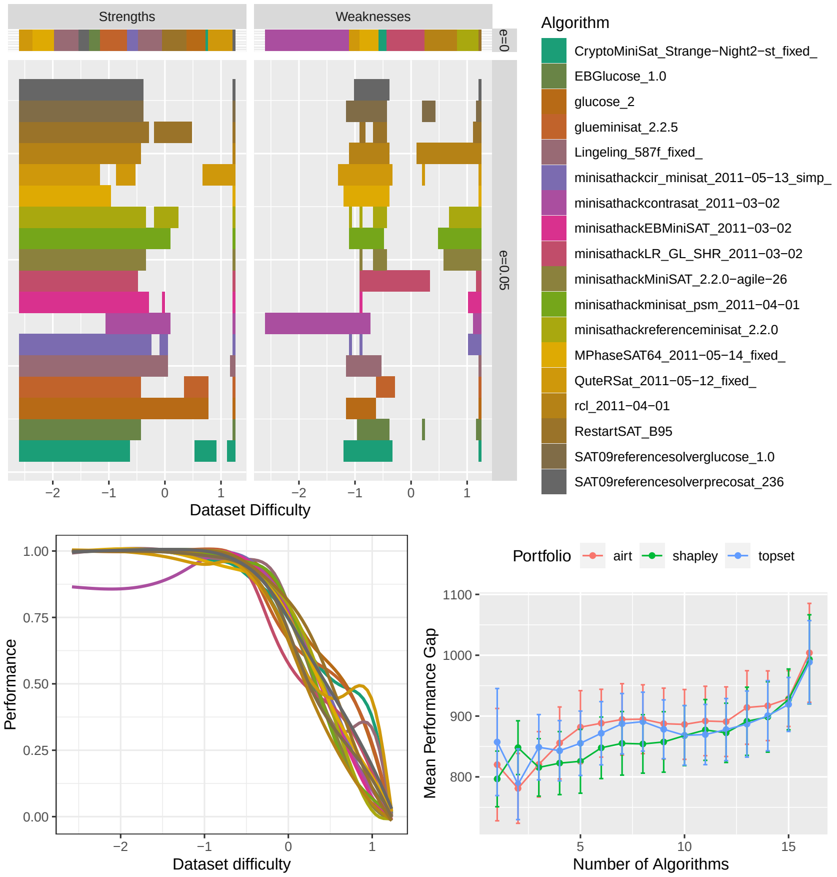


Figure 29: Strengths and weaknesses, latent trait curves and 10-fold CV portfolio comparison for SAT11_INDU scenario..

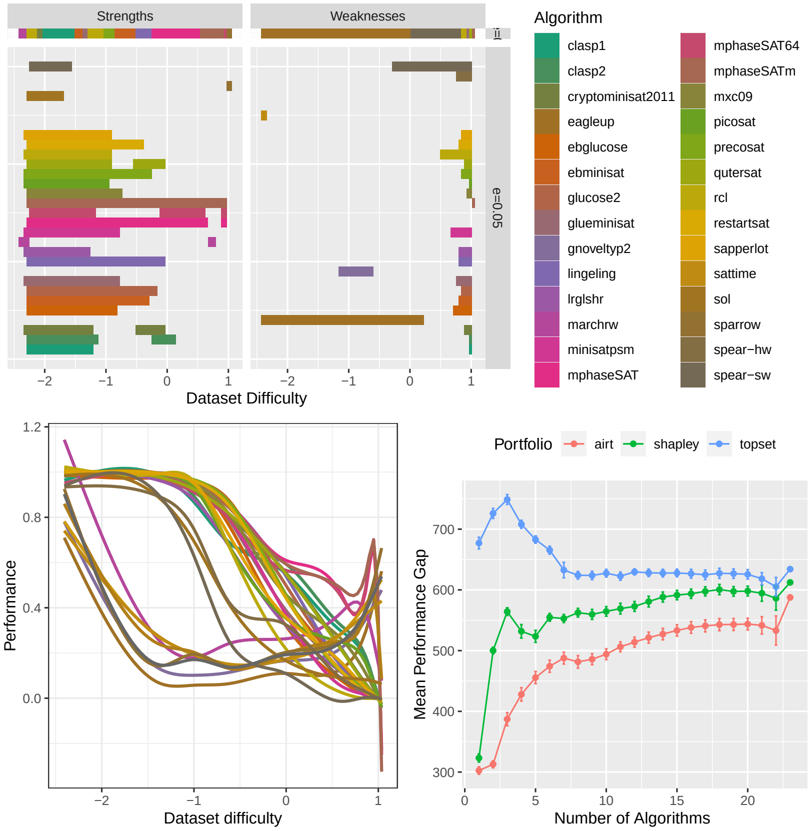


Figure 30: Strengths and weaknesses, latent trait curves and 10-fold CV portfolio comparison for SAT12_ALL scenario.

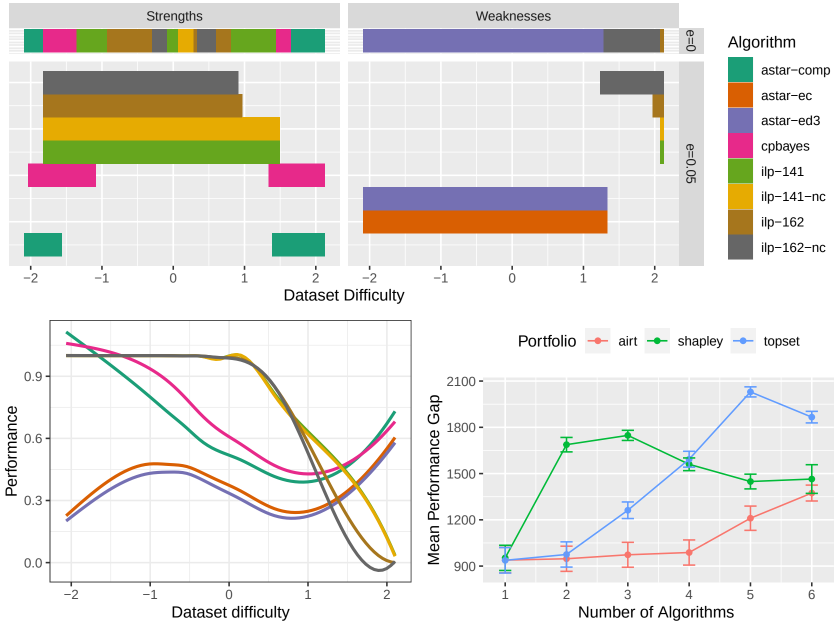


Figure 31: Strengths and weaknesses, latent trait curves and 10-fold CV portfolio comparison for BNSL_2016 scenario..

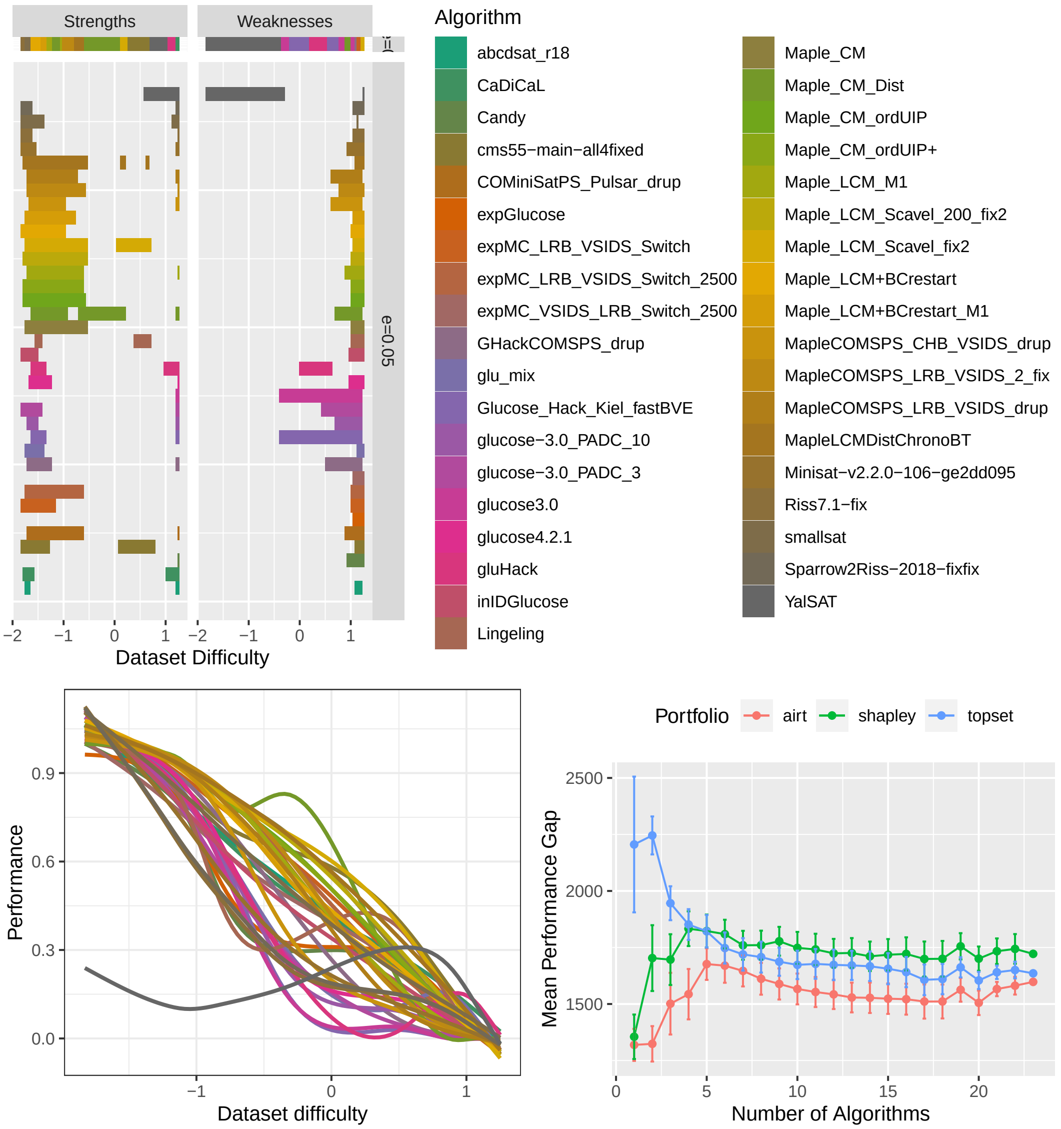


Figure 32: Strengths and weaknesses, latent trait curves and 10-fold CV portfolio comparison for SAT18_EXP_ALGO scenario..