Universiteit Leiden
The Netherlands

Bachelor DSAI

Comparing traditional and deep learning-based meta-features for algorithm selection

Khayri Hamza

Supervisors:
Dr. J.N. van Rijn
Dr. Edesio Alcobaça (Universidade de São Paulo, Brazil)

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

20/01/2026

Abstract

The performance of meta-learning systems for algorithm selection relies on the quality of the meta-features used. While deep learning–based meta-features have proven effective for an adjacent meta-learning task, hyperparameter optimisation, there remains scope for further exploration of their effectiveness for algorithm selection specifically. This thesis evaluates three meta-feature types—traditional, deep learning, and combined—for algorithm selection. We developed a tool that systematically leverages OpenML resources to extract datasets and meta-features, and to obtain deep learning–based dataset representations using Dataset2Vec, an existing deep learning meta-feature extraction tool, enabling reproducible experiments. The results indicate that traditional meta-features outperform deep learning–based representations, while the combined approach shows intermediate performance. These findings suggest that current deep learning–based meta-features do not yet capture the dataset characteristics most relevant for algorithm selection.

Contents

1 Introduction

Machine learning has influenced how we address various problems. From the potential of self-driving cars [PKP22] to reframing our understanding of sports [Fuj25] to support clinical detection [ZSW23], algorithms have become essential to learning from the data around us. With such diverse applications, no single algorithm can excel at everything [BvRSV22].For example, algorithms that excel in image recognition may perform poorly on text classification, while methods optimised for high-dimensional data may struggle with small sample sizes [Bis06]. This reveals the challenge — how to choose the optimal algorithm systematically without relying on exhaustive trial-and-error approaches.

Different algorithms perform well on different problems, and selecting a suboptimal algorithm can lead to mediocre performance, making the choice of algorithm essential for achieving acceptable results. Rice formalised this challenge into the algorithm selection framework, which treats algorithm selection as a learning problem that can be addressed through meta-learning methods [Ric76].

Meta-learning addresses the algorithm selection problem framework [GC08] by learning from multiple problems to determine which problems specific algorithms are suited to, leveraging prior experience for more efficient and effective generalisation. Meta-learning systems aim to learn the relationship between problem characteristics and algorithm performance across different tasks. However, the problem characteristics (meta-features) can be defined by various approaches and complexities. The quality of meta-features is crucial to determining meta-model effectiveness [BvRSV22, RGS+19], and there are different ways to create expressive meta-features efficiently.

Different approaches exist for meta-feature extraction, ranging from statistical measures [BvRSV22], to landmarkers [BGC00], to deep learning–based dataset representations, such as those produced by Dataset2Vec [JSTG21]. While traditional meta-features have been proven to be effective, there remains scope for developing meta-features better suited to algorithm selection. Deep learning approaches have the potential to capture complex dataset relationships and have shown promising results for hyperparameter optimisation; however, the effect on algorithm selection, an adjacent meta-learning task, remains understudied.

This study addresses the following research question:

- How effective are deep learning extracted meta-features compared to traditional meta-features for algorithm selection in classification tasks?

The key research question is guided by the sub-research questions:

- To what extent do Dataset2Vec representations capture underlying structure relevant for algorithm selection?
- How does varying base learners affect the performance of meta-models trained on different types of meta-features?

This study leverages the extensive OpenML platform [VvRBT13] to access the rich dataset repository and contributions of the broader machine learning community [BCD+25]. We developed a tool to systematically extract traditional meta-features and Dataset2Vec representations to construct meta-learning systems for algorithm selection, to conduct experiments in order to evaluate the

performance to answer the research questions. We further investigate how well deep learning meta-features capture dataset characteristics and examine how changes in the algorithm space affect meta-model performance.

Our contributions are the following:

- Systematically comparing meta-models trained on three different sets of meta-features — traditional meta-features, deep learning extracted meta-features, and a combined set of meta-features. We perform this evaluation on 2 benchmark suites across various subsets of algorithms
- Leveraging the existing infrastructure of the OpenML platform that can be scaled up across a diverse set of problems and tasks

Together, these contributions provide a reproducible framework and further insights into designing accessible and efficient meta-learning systems for algorithm selection.

2 Background

This section establishes the following core concepts that will be addressed throughout the study: The algorithm selection problem, meta-learning for algorithm selection, meta-features, OpenML, and PyMFE.

2.1 The Algorithm Selection Problem

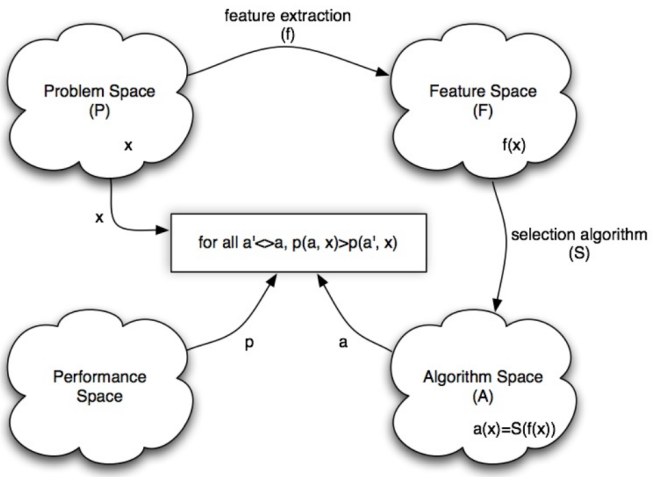


Figure 1: The algorithm selection framework, formalising the mapping from problem instances and their features to an algorithm space defined by expected performance. Taken from [GC08], p. 18

Different algorithms excel under different conditions; no single method dominates universally across tasks [BvRSV22]. For example, decision trees might perform well on tabular datasets with clear hierarchical splits, whereas support vector machines historically succeed on high-dimensional

text data. Similarly, k-nearest neighbours have been speculated to be competitive on small, low-dimensional problems but scale poorly as datasets grow [Qui86, CV95, Bre01]. This variability defines the algorithm selection problem: the challenge of choosing the best-performing algorithm for a specific problem instance.

Rice (1976) formalised this challenge through his framework [Ric76] (Figure 1), which describes the algorithm selection problem in terms of four spaces:

- the **problem space** (P), which contains all possible problem instances;
- the **feature space** (F), which represents descriptive characteristics of problems (e.g., number of features, class entropy, correlations);
- the **algorithm space** (A), which enumerates the available algorithms; and
- the **performance space** (Π), which evaluates outcomes using a metric such as accuracy or runtime.

In practice, researchers often approach dataset-level algorithm selection by testing a range of candidate algorithms on the same dataset and choosing the one with the highest validation performance. This trial-and-error approach of conventional algorithm selection is computationally expensive as the number of datasets and algorithms increases, and it does not exploit structural similarities between problems [GC08, BvRSV22].

These limitations motivate the development of meta-learning systems that learn from prior experience. Instead of retraining every algorithm for each new task, such systems extract features from datasets (e.g., number of classes or skewness) and construct a meta-model that predicts which algorithm will perform best. In this way, meta-learning reduces computational costs and provides effective decision support to practitioners, enabling the technology to scale beyond exhaustive search.

2.2 Meta-Learning for Algorithm Selection

Meta-learning addresses algorithm selection by learning from prior experiences across different problems [Van18].

A meta-learning system for algorithm selection (Figure 2) consists of two processes — a learning subsystem and a meta-model [CGL23]. The learning subsystem generates meta-data by training ML algorithms (base learners) on a set of early encountered problems and collecting their performances. In parallel, descriptive characteristics (meta-features) of each dataset are extracted. Together, these produce meta-data pairs connecting meta-features to base learner performances. The meta-model is then trained on the meta-data to learn the relationship between dataset characteristics and algorithm choice.

Meta-features, while broadly defined as dataset properties, are a crucial component of meta-learning systems [BvRSV22]. They have to effectively characterise a dataset to ensure clear mapping to algorithm performance. Hence, the quality of meta-features is paramount to the effectiveness of meta-learning for algorithm selection.

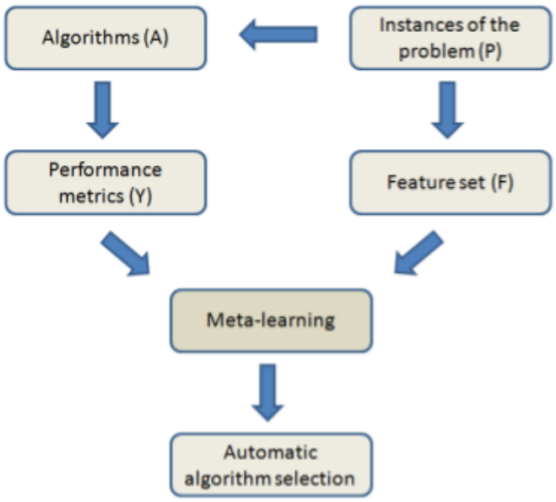


Figure 2: Pipeline of a meta-learning system for algorithm selection, illustrating the learning subsystem and the resulting meta-model. Taken from Carvalho et al. [CGL23], p. 2.

2.3 Meta-Features

Meta-features are descriptors that capture properties of datasets beyond individual instances [RGS+19]. This allows consistent comparison across datasets, enabling meta-learning systems to learn across problems.

2.3.1 Traditional Meta-Features

Traditional meta-features are hand-crafted descriptors of datasets. They are widely used in meta-learning for algorithm selection and can be categorised into the following:

- **Simple** – Basic dataset descriptors such as number of instances, number of features, or proportion of missing values [Aha92, MST94].
- **Statistical** – Capture distributional properties of features, such as mean, variance, skewness, or correlation.
- **Model-Based** – Characterise datasets using parameters or diagnostics derived from simple models trained on the data (e.g., decision tree depth, number of rules) [HK01].
- **Information-Theoretic** – Measure information content (e.g., class entropy, mutual information between features and labels, or noise levels).
- **Landmarking** – Use the performance of fast, inexpensive "landmarker" algorithms (e.g., 1-NN, Naïve Bayes) as indicators of dataset properties [PBGC00, BGC00].
- **Complexity** – Quantify structural difficulty of the learning task, such as feature overlap between classes or linear separability [Smi08].

Traditional meta-features can be extracted through the OpenML API when computed and available or directly from datasets using the `PyMFE` library. Using traditional meta-features involves manual engineering across tasks [ASR+20]. This limits the scalability of the traditional meta-feature approach to larger sets of problems [BvRSV22].

2.3.2 Deep-Learning Feature Extraction

Deep learning approaches for meta-feature extraction learn representations directly from the data [JSTG21]. They compress the complex relationships between features and target into a vector representation, creating unique 'fingerprints' of each dataset.

Dataset2Vec builds on this idea by representing tasks as hierarchical sets of predictor-target pairs. The pairs are passed through a DeepSet architecture [ZKR+17]. Dataset2Vec uses the auxiliary task of dataset similarity to train. In doing so, it develops representations that can capture what makes datasets unique and comparable.

Dataset2Vec meets the following desiderata that are essential to an effective meta-feature extraction method [JSTG21]:

- Schema Agnosticism - extracting meta-features for datasets with varying schema
- Expressivity - extracting meta-features for meta-tasks of varying complexity
- Scalability - extracting meta-features quickly for new datasets without requiring training on new tasks
- Correlation - meta-features should correlate well with meta-targets

Dataset2Vec fulfils the desired criteria for a descriptive meta-feature extractor and displays positive results in the meta-learning task of hyperparameter optimisation. However, its suitability for a different meta-learning task, like algorithm selection, remains unexplored. Hyper-parameter optimisation is a more nuanced description within single algorithm paradigms, whereas algorithm selection distinguishes between fundamentally different algorithm approaches. This thesis evaluates whether the representations created by Dataset2Vec are relevant and effective for algorithm selection.

2.4 OpenML

OpenML is an open platform designed to make machine learning research transparent and reproducible [VvRBT13, BCD+25]. It functions as a dataset repository enriched with meta-features, as well as a collaborative environment for sharing experiments.

Key functionalities include:

- Rich datasets and meta-features – diverse datasets and computed meta-features support large-scale meta-learning [RGS+19].
- Benchmark Suites – Curated collections of tasks for consistent and reproducible comparison of algorithms [BCF+21].
- Runs – enables reuse and comparison of other experiments, building directly on the work of others.

- Accessible infrastructure – APIs and integrated libraries make it possible to use in experiments [CGL23].

While OpenML provides extensive resources, meta-feature coverage varies across datasets [CGL23], and the platform requires manual curation to construct meta-learning pipelines for algorithm selection. Nevertheless, OpenML's scale and collaborative infrastructure make it a valuable foundation for systematic meta-learning research.

2.5 PyMFE Library

The Python Meta-Feature Extractor (PyMFE) is an open-source library developed to standardise and systematise meta-feature extraction for meta-learning research [ASR+20]. PyMFE addresses the reproducibility challenges in meta-learning by providing a comprehensive framework that implements over 90 characterisation measures across six meta-feature categories: simple, statistical, information-theoretic, model-based, landmarking, and complexity measures.

The library follows the formal meta-feature definition proposed by Rivolli et al. [RGS+19]: $f(D) = \sigma(m(D, h_m), h_\sigma)$, where m is a characterisation measure that computes descriptive values from the dataset D, σ is a summarisation function that aggregates these values (e.g., mean, standard deviation), and h_m, h_σ are hyperparameters. For example, computing feature correlations (m) and taking their mean (σ) yields a single meta-feature describing average feature correlation.

PyMFE provides a sklearn-inspired interface with two primary methods: `fit()` computes necessary data transformations and pre-computations, while `extract()` applies characterisation measures followed by summarisation to return the final meta-features. The library supports flexible meta-feature selection, allowing users to extract individual measures, entire categories, or custom combinations.

The implementation emphasises reproducibility through comprehensive unit testing (over 90% code coverage), adherence to Python coding standards, and extensive documentation. PyMFE leverages robust scientific computing libraries, including NumPy, scikit-learn, and SciPy, to ensure reliable computations.

3 Methods

This section describes the development of a meta-learning tool for algorithm selection. The tool consists of four main components: a dataset collection module that interfaces with OpenML to retrieve benchmark tasks, a meta-feature extraction pipeline that generates multiple types of meta-features, a performance aggregation system that processes algorithm execution results, and a meta-model training system that learns algorithm selection relationships.

[1]

[1] Our full code is available here: https://github.com/khayhamz31/algoselectionpipeline

3.1 Dataset Collection

The dataset collection module provides a systematic interface to OpenML benchmark suites [VvRBT13, BCF+21]. The module retrieves datasets from curated collections and applies preprocessing transformations to ensure consistent formatting across different data sources.

The preprocessing pipeline handles common data quality issues, including missing values, mixed data types, and inconsistent encodings [GLH15]. This standardisation enables reliable meta-feature extraction and ensures datasets are in a suitable format for downstream analysis components.

3.2 Meta-Feature Extraction

The meta-feature extraction pipeline generates three types of dataset representations to enable systematic comparison of different characterisation approaches.

3.2.1 Traditional Meta-Features

The traditional meta-features component extracts pre-computed traditional meta-features through the OpenML API [VvRBT13]. The component implements systematic missing value handling procedures to ensure complete feature vectors while assessing feature reliability based on data availability patterns.

3.2.2 Deep Learning Meta-Features

The deep learning component implements Dataset2Vec to generate fixed-dimensional vector representations for each dataset [JSTG21]. The component processes datasets through the pre-trained model to produce dense vector representations that capture complex dataset characteristics.

3.2.3 Hybrid Meta-Features

The hybrid component combines traditional and deep learning representations through vector concatenation. This approach enables evaluation of whether different meta-feature types provide complementary information for algorithm selection.

3.3 Performance Aggregation

The performance aggregation component processes OpenML execution records to determine optimal algorithms for each dataset [BCD+25]. OpenML maintains extensive records of algorithm executions across multiple datasets, providing empirical performance data without requiring independent computational resources.

The component addresses the complexity of OpenML's flow system, which bundles core algorithms with preprocessing steps and hyperparameter configurations [VvRBT13]. The aggregation process maps these detailed implementations back to core algorithmic approaches while handling performance measurements per underlying dataset combination.

The system generates ground-truth algorithm performance rankings that serve as targets for meta-learning, enabling the tool to learn which algorithms perform optimally under different dataset conditions.

3.4 Meta-Model Training

The meta-model training stage learns the relationship between dataset characteristics and algorithm performance from the constructed meta-data. Although the same meta-data underlie each approach, the training procedure differs depending on how algorithm performances are represented and which type of predictive model is used. The distinction lies in whether the meta-model predicts a discrete algorithm choice or a full vector of performance values.

3.4.1 Meta-Classifier

In the meta-classifier framework, the performance matrix is reduced to a single label per dataset by selecting the best-performing algorithm. The meta-features serve as input features and the best algorithm acts as the target. A supervised classifier is then trained to map meta-features to algorithm labels, producing a model capable of predicting the most suitable algorithm for unseen datasets without exhaustive benchmarking.

3.4.2 Meta-Regressor

The meta-regressor framework retains the full performance vector for each dataset rather than collapsing it to a single label. The meta-features are used to predict the continuous performance values of all candidate algorithms through multi-output regression. The resulting regressor predicts the performance of each algorithm, and the recommended algorithm can be inferred by selecting the maximum.

3.4.3 Meta-Model Comparison

The two approaches offer complementary perspectives. The meta-classifier focuses on learning decision boundaries between algorithms, evaluating whether the meta-features contain enough information to identify the single best choice. In contrast, the meta-regressor assesses whether the same meta-features capture finer-grained performance differences across algorithms. Comparing the two reveals whether the available representations support both coarse algorithm selection and detailed performance modelling.

4 Experimental Setup

This section describes the implementation of the tool and design choices for dataset selection, preprocessing, meta-feature extraction, performance aggregation, and evaluation.

4.1 Dataset Selection and Preprocessing

OpenML contains 24157 datasets, with 6279 of them verified. It is a rich resource that provides dataset information [VvRBT13, BCD+25]. There are 118 benchmark suites, which are curated

collections of tasks, that allow for standardised, reproducible, and consistent comparisons [BCF+21].

We selected the following benchmark suites for this study:

- **OpenML-CC18 (id=99)** – Provides a curated collection of 72 medium-sized datasets that are practical to use, standardised and valid. It already serves as a widely adopted benchmark that enables reproducible comparisons across studies [BCF+21].
- **OpenML100-friendly (id=225)** - A filtered version of OpenML-100 that contains 54 tasks. All the datasets in the task have no missing values and only numerical features [CGL23]. It enables broad algorithm comparison without specialised preprocessing, while maintaining diverse datasets.

We selected these benchmark suites as two of the three active benchmark suites on OpenML, and they both contain classification tasks. These suites provide complementary perspectives - OpenML-CC18 represents diverse real-world classification tasks while OpenML100-friendly enables controlled comparison without preprocessing complexity, together testing meta-feature effectiveness across different dataset conditions.

4.2 Meta-Feature Extraction

We produced three representations: traditional meta-features, Dataset2Vec representations, and a hybrid concatenation of both types of meta-features. This produced 3 distinct feature matrices that served as alternative inputs for training and evaluating meta-model performance.

4.2.1 Traditional Meta-Features

OpenML provides pre-computed traditional meta-features for datasets and can be accessed using the OpenML API [BvRSV22, RGS+19]. We systematically downloaded them and organised for the meta-model to be trained.

We applied the following preprocessing steps:

- Identifying and reporting overall missing values across all meta-features, to monitor data completeness and guide subsequent imputation choices,
- Removing meta-features that are empty across all datasets, since they provide no information for training,
- Assessing feature reliability by categorising meta-features according to missingness percentage, to distinguish consistently reliable features from those with sporadic availability.
- Imputing missing values in the remaining meta-features using KNN imputation ($k = 5$), which preserves local structure and leverages similarities between datasets.

4.2.2 Deep learning Meta-Features

We applied Dataset2Vec to each dataset in the benchmark suites to generate deep-learning meta-features [JSTG21, ZKR+17]. Dataset2Vec processes a dataset as a set of feature-instance pairs and outputs a fixed-length representation that captures higher-order patterns such as feature

interactions and distributional structure, which are not easily represented by traditional statistical descriptors.

Before generating deep-learning meta-features with Dataset2Vec, we preprocessed each dataset to ensure compatibility with the model. We used the OpenML API to download and store datasets systematically.

Each dataset undergoes the following preprocessing steps:

- Separate features and targets with no column headers for meta-feature extraction
- Remove features with more than 70% missing values to ensure sufficient data quality
- Remove samples with more than 50% missing values
- Impute categorical features with the most frequent value
- Impute numerical features with KNN ($k = 5$), or median imputation if too few samples are available to preserve data structure
- Scale numerical features to [0,1] using Min–Max scaling
- Encode categorical features using one-hot encoding
- Label-encode the target variable into integer classes

For each dataset, we extracted a 32-dimensional representation from the pre-trained Dataset2Vec model and consolidated the results into a feature matrix consistent with the traditional meta-features. This alignment ensured that both traditional and deep-learning representations could serve as interchangeable inputs for meta-model training and direct performance comparison.

4.2.3 Hybrid Meta-Features

We combined traditional and deep learning meta-features through vector concatenation to assess whether these different representation types provide complementary information for algorithm selection. This hybrid approach enables direct evaluation of whether learned representations enhance the discriminative power of established statistical descriptors or whether the approaches capture overlapping dataset characteristics.

4.3 Performance Data Extraction

As outlined in Section 3.3, we aggregated algorithm evaluation metrics across tasks in both benchmark suites to determine base learner performance. This required extracting and processing run-level information from OpenML.

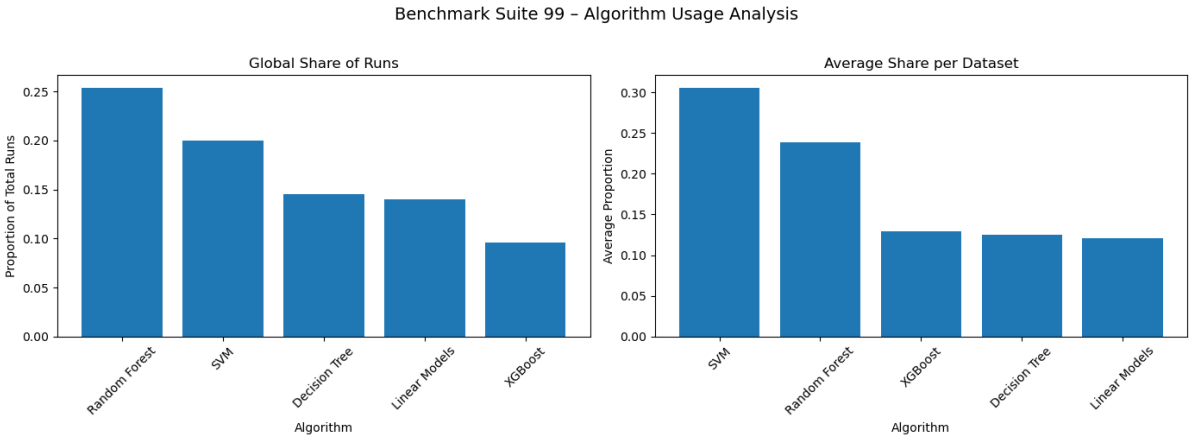
4.3.1 OpenML Runs

In OpenML, runs represent individual executions of algorithms and processes on a specific problem. They link a task, which consists of the dataset, problem, and evaluation, with a flow, which consists of an implementation of an algorithm and pipeline [VvRBT13, BCD+25]. Researchers execute runs locally and upload the task, flow and resulting performance metrics to the OpenML platform. This

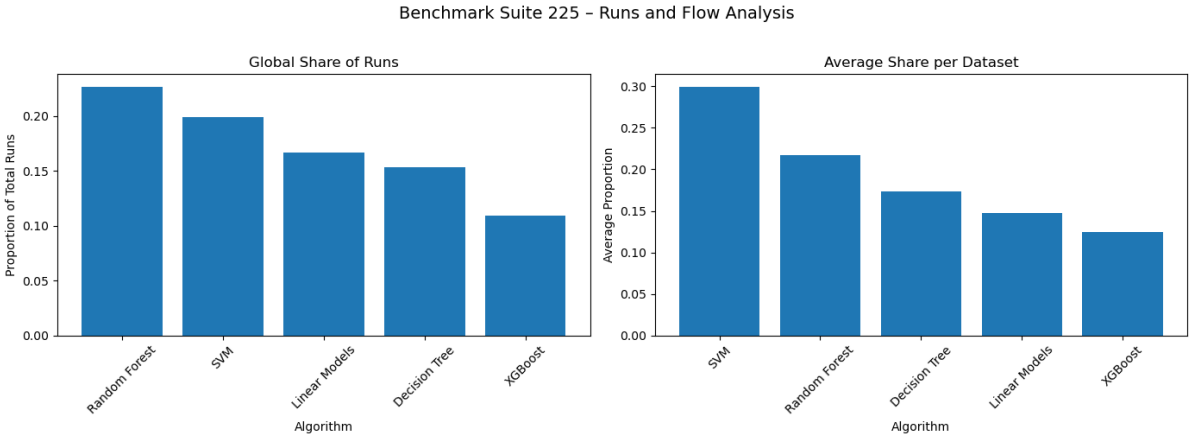
makes runs a valuable resource, as many algorithm configurations and executions are accessible without needing to individually run.

4.3.2 Flows

Flows represent specific configurations of algorithms, which can include preprocessing methods (e.g. imputation methods and encoding methods) and hyperparameter configurations [LBNA+03, KTK00, BK01]. While this level of detail is necessary for transparency and reproducibility, it complicates the experiment of algorithm selection that focuses on the core algorithm and not its entire implementation.



(a) OpenML-CC18



(b) OpenML100-friendly

Figure 3: Distribution of core algorithms across two benchmark suites, OpenML-CC18 and OpenML100-friendly.

Preliminary analysis of all the runs across benchmark suite OpenML-CC18 and OpenML100-friendly (Fig. 3) shows that the most common and well-represented core algorithms are: Support Vector Machine, Random Forest, Decision Tree, XGBoost, and Linear Models (such as logistic regression and permutations of it) [CV95, Bre01, Qui86, CG16]. The algorithms are both methodologically

diverse and have sufficient coverage across datasets in both benchmark suites. Hence, we selected these five algorithms as the set of base learners for this thesis.

We created a static mapping of OpenML flows to five base learner categories through a one-time offline annotation process. We leveraged a large language model to assist with semantic classification, handling the diverse naming conventions in the metadata and filtering out hyperparameters irrelevant to algorithm selection. After manual review, this mapping was stored as a lookup table and used consistently across all experiments, ensuring the pipeline itself has no dependency on external services.

4.3.3 Performance Metrics

We chose accuracy as the performance measure because it is widely used, easy to interpret, and well-suited for classification tasks. It provides a consistent baseline across datasets. Formally, accuracy is defined as:

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(\hat{y}_i = y_i)\,, \tag{1}$$

where N is the number of instances, y_i is the true class label, $\hat{y}_i$ is the predicted class label, and $\mathbb{1}(\cdot)$ is the indicator function that equals 1 if the prediction is correct and 0 otherwise.

	OpenML-CC18		OpenML100-friendly	
Processing Step	% Removed	Runs Removed	% Removed	Runs Removed
Filtering core algorithms	22.0	832,306	18.0	647,440
Sampling runs	99.5	2,617,612	99.4	2,936,050

Table 1: Global reduction in run data at each processing step.

We only focus on the runs that belong to the current base learner set; we filtered out runs corresponding to other algorithms (Table 1). Even after the initial filtering, the number of runs is still very large, making it computationally expensive to extract all evaluation metrics from the OpenML API. To reduce while maintaining diversity, we sample 50 runs each per dataset-algorithm pair (Table 1). From the 50 runs, we selected the median of the top 10 highest-performing runs. This approach ensures good performance while accounting for variance and outliers that could skew results from a single best run.

4.4 Meta-Classifier Training and Evaluation

With the meta-features extracted, we trained and evaluated supervised meta-classifiers to compare how effective traditional, deep, and hybrid meta-features support algorithm selection.

4.4.1 Meta-Classifier Architecture

The meta-classifier learns the relationship between meta-features and the best-performing base learner. We adopted a supervised learning approach, with meta-features as inputs and the encoded best learner as the classification target.

In the context of the meta-learning task of algorithm selection, the classifier learns the decision boundaries that separate regions of the meta-feature space where different algorithms are optimal. The meta-features should contain sufficient discriminative power to determine the optimal algorithm.

We selected the random forest classifier as the meta-classifier due to its balance between computational efficiency and modelling sophistication [Bre01]. Random forests can train quickly, handle heterogeneous feature spaces, capture non-linear relationships, and provide feature importance estimates, while remaining relatively robust to overfitting.

4.4.2 Evaluation Protocol

For each benchmark suite, we computed a majority-class baseline by selecting the base learner most frequently optimal across datasets [Koh95]. This baseline provides a reference point against which meta-model performance is evaluated.

We assessed predictive performance using Leave-One-Out Cross-Validation (LOO-CV) across the datasets in each benchmark suite [Sto74]. Each dataset was iteratively held out as a test case while the others were used for training, ensuring evaluation reflects generalisation to unseen datasets — the central objective in algorithm selection. LOO-CV is particularly suitable when the number of datasets is limited, as with our benchmark suites.

To account for model variance, we repeated training and evaluation for 10 independent runs with different random seeds. We report results as median accuracy and standard deviation, capturing both central tendency and variability. This enables robust comparison of meta-models trained with different meta-feature representations.

4.5 Meta-Regressor Training and Evaluation

With the meta-features extracted, we trained and evaluated supervised meta-regressors to compare how effective traditional, deep, and hybrid meta-features support algorithm selection.

4.5.1 Meta-Regressor Architecture

The meta-regressors learns the relationship between meta-features and the performances of all the base learners. We adopted a supervised learning approach, with meta-features as inputs and base learner performances as the target.

In comparison to the meta-classifier, the meta-regressor models the algorithm selection problem by predicting continuous base learner performance. This approach captures the competitiveness between algorithms, representing more nuanced performance that reducing to discrete labels as the meta-classifier does is unable to. The most suitable algorithm is then chosen by selecting the algorithm with the highest predicted performance.

4.5.2 Evaluation Protocol

We evaluated the meta-regressor using Leave-One-Out Cross-Validation (LOO-CV) across datasets [Sto74]. In each repetition, the regressor was trained on all but one dataset and used to predict the algorithm performances for the held-out dataset to generalise across datasets.

To provide a baseline to compare the performance of the regressor, we computed a mean-predictor, which predicts the average performance of each algorithm across the training datasets. This provides a simple baseline for assessing whether the meta-regressor captures meaningful performance variation.

Model quality was quantified using standard regression metrics—Mean Absolute Error (MAE), Mean Squared Error (MSE), and coefficient of determination (R^2)—evaluated over the predicted and actual algorithm performance matrices. To capture variability due to model stochasticity, the full LOO-CV process was repeated for 10 runs with different random seeds, and we report the aggregated mean and standard deviation for each metric.

4.6 Dataset2Vec Representation Analysis

Before evaluating Dataset2Vec as a meta-feature representation for algorithm selection, we examine whether the representations produced by Dataset2Vec capture meaningful aspects of dataset structure. In particular, we consider whether the embeddings distinguish between datasets with similar characteristics and those that differ, providing an initial validation of their suitability as meta-features.

We assessed the quality of the Dataset2Vec representations using a benchmark consisting of 2000 synthetic datasets comprising circles, blobs and moons, each produced with randomised noise levels, transformations and sample sizes. This ensures that there are distinct differences within each geometric family for Dataset2Vec to process.

We visualised the representation space using t-SNE to assess whether datasets from the same geometric family cluster together. To quantify this structure, we computed cosine distances between all dataset pairs and compared intra-class distances (within the same family) to inter-class distances (between different families). The inter/intra distance ratio measures separability: values above 1 indicate that Dataset2Vec maps similar datasets closer together than dissimilar ones.

This analysis evaluates whether the learned representations reflect the underlying dataset structure relevant for downstream meta-learning tasks.

4.7 Algorithm Subsets

To analyse algorithm selection behaviour under different experimental conditions, we evaluate meta-models on restricted subsets of algorithms rather than only on the full set of base learners. The subset analysis is operationalised along two complementary dimensions: algorithm structure and empirical performance distribution.

4.7.1 Algorithm Structure

Algorithms often share common inductive biases, making them harder to distinguish. To test whether meta-features capture such fine-grained structural differences, we defined three structural subsets:

Tree Algorithms {Decision Tree, Random Forest, XGBoost}: We tested whether meta-models could discriminate within tree-based methods, from simple decision trees to ensemble bagging and boosting, evaluating whether meta-features reflect increasing algorithmic sophistication within one family.

Ensemble Algorithms {Random Forest, XGBoost}: We tested whether meta-features distinguish ensemble methods that rely on different aggregation strategies — parallel bagging versus sequential boosting — despite both building multiple models.

Non-Ensemble Algorithms {Decision Tree, SVM, Linear Models}: We tested whether meta-features could separate single model paradigms with fundamentally different learning assumptions, providing a baseline of maximum algorithmic diversity.

4.7.2 Empirical Performance Distribution

Beyond structural design, algorithm strength can also be assessed empirically by how well algorithms perform across datasets. As a preliminary analysis, we summarise the average accuracy of the algorithms across the benchmark suites. This summary guides the selection of algorithm pairs for comparison.

Benchmark Suite	Algorithm	Average Accuracy
OpenML-CC18 (99)	Random Forest	0.864 ± 0.146
	Support Vector Machine	0.860 ± 0.161
	Linear Models	0.824 ± 0.164
	XGBoost	0.811 ± 0.184
	Decision Tree	0.799 ± 0.155
OpenML100-friendly (225)	Random Forest	0.879 ± 0.107
	Support Vector Machine	0.877 ± 0.116
	Linear Models	0.825 ± 0.177
	Decision Tree	0.806 ± 0.143
	XGBoost	0.765 ± 0.238

Table 2: Average aggregated accuracy per algorithm across benchmark suites OpenML-CC18 (ID 99) and OpenML100-friendly (ID 225). Algorithms are ranked by descending average accuracy within each suite.

Based on the average accuracy rankings in Table 2, we define three comparative settings to analyse algorithm selection under different empirical performance regimes.

1. Strong vs. strong: This setting compares the two highest-ranked algorithms within a benchmark suite. It represents a scenario in which both algorithms perform well on average, and selection

depends on distinguishing subtle dataset characteristics that favour one strong performer over another.

2. Weak vs. weak: This setting compares the two lowest-ranked algorithms within a benchmark suite. It reflects situations where neither algorithm performs particularly well on average, testing whether the meta-model can identify the comparatively better option among weaker candidates.

3. Strong vs. weak: This setting compares the highest-ranked algorithm with the lowest-ranked algorithm within a benchmark suite. It represents a clear performance contrast, testing whether the meta-model can consistently identify the stronger algorithm while accounting for occasional exceptions.

4.8 Configurations and parameters

Table 3 consolidates all the configurations and parameters in the methodology into a single table for reference.

5 Results

In this section, we present the results of our experiment by addressing the research questions through a comparison of meta-learning approaches that rely on traditional, deep learning and hybrid meta-feature representations, as well as further analysis of Dataset2Vec.

5.1 Main results across all base learners

This section presents the results of the main experiment, which evaluates the relative effectiveness of each meta-feature type for algorithm selection across the full set of base learners.

5.1.1 Meta-classifier results

This section reports the performance of the meta-classifier for each meta-feature representation.

Figure 4 and Table 4 establish a consistent performance ranking across both benchmark suites. Traditional meta-features achieve the highest accuracy, followed by the hybrid approach, and Dataset2Vec performs the poorest. Notably, for the OpenML-CC18 benchmark suite, it performs worse than the baseline, and for the OpenML100-friendly benchmark suite it barely exceeds the baseline, indicating minimal learning. This performance gap is consistent, and the low standard deviations indicate that the differences are stable across experimental results.

5.1.2 Meta-regressor results

In this section, we evaluate the meta-regressor purely as a regression model, assessing its ability to predict base-learner performance using continuous regression metrics.

Table 5 reveals a consistent performance hierarchy across both benchmark suites when predicting algorithm performance. Traditional meta-features demonstrate the best predictive accuracy, achieving the lowest MAE (0.0720 for OpenML-CC18; 0.0562 for OpenML100-friendly) and highest

Table 3: Consolidated overview of configurations and parameters used in the methodology.

Component	**Configuration / Parameters**
Benchmark Suites	OpenML-CC18 (72 datasets); OpenML100-friendly (54 datasets)
Dataset Preprocessing (Dataset2Vec)	Applied *only for Dataset2Vec input construction*: remove features >70% missing; remove samples >50% missing; KNN imputation ($k = 5$) or median imputation; Min–Max scaling; one-hot encoding for categorical features; label-encoding for targets
Meta-Features	Traditional (OpenML qualities, KNN-imputed); Dataset2Vec; Hybrid (concatenation of traditional + D2V)
Meta-Targets	Best algorithm by accuracy (classification); accuracy vector over algorithms (regression)
Base Learners	SVM, Random Forest, Decision Tree, XGBoost, Linear Models
Run Filtering	Only runs corresponding to the selected base learners were retained (Table 1); 50 sampled runs per dataset–algorithm pair
Performance Aggregation	Median of top 10 accuracies per dataset–algorithm pair
Meta-Classifier Model	Random Forest Classifier
Meta-Regressor Model	Random Forest Regressor (multi-output)
Baseline	Majority-class baseline (meta-classification); mean-predictor baseline (meta-regression)
Evaluation	Leave-One-Out cross-validation across datasets (10 random-seed repetitions)
Algorithm Subsets	**Structure-based**: Tree Family {DT, RF, XGB}; Ensemble {RF, XGB}; Non-Ensemble {DT, SVM, Linear}. **Empirical performance distribution-based**: Strong vs. Strong, Strong vs. Weak, Weak vs. Weak, derived from empirical win rates (Table 2).
Representation Analysis	t-SNE visualisation of Dataset2Vec embeddings; quantitative evaluation of separability using intra-/inter-class cosine distances
Results Reporting	Median accuracy (classifier) and median MAE/MSE/R^2 (regressor) with standard deviations across 10 repetitions

R^2 values (0.511 and 0.691, respectively). The hybrid approach provides marginal gains over traditional features on OpenML100-friendly ($R^2 = 0.719$ vs. 0.691), but offers no advantage on OpenML-CC18. Dataset2Vec embeddings consistently underperform, yielding substantially higher prediction errors and poor variance explanation. Particularly concerning is Dataset2Vec's negative R^2 on OpenML-CC18 (-0.222 ± 0.008), indicating predictions worse than a naive mean baseline, while on OpenML100-friendly it achieves only minimal positive R^2 (0.140), suggesting limited learning of algorithm performance patterns. The low standard deviations across all metrics (< 0.001 for MAE and MSE) indicate these performance differences are robust and stable across repeated evaluations.

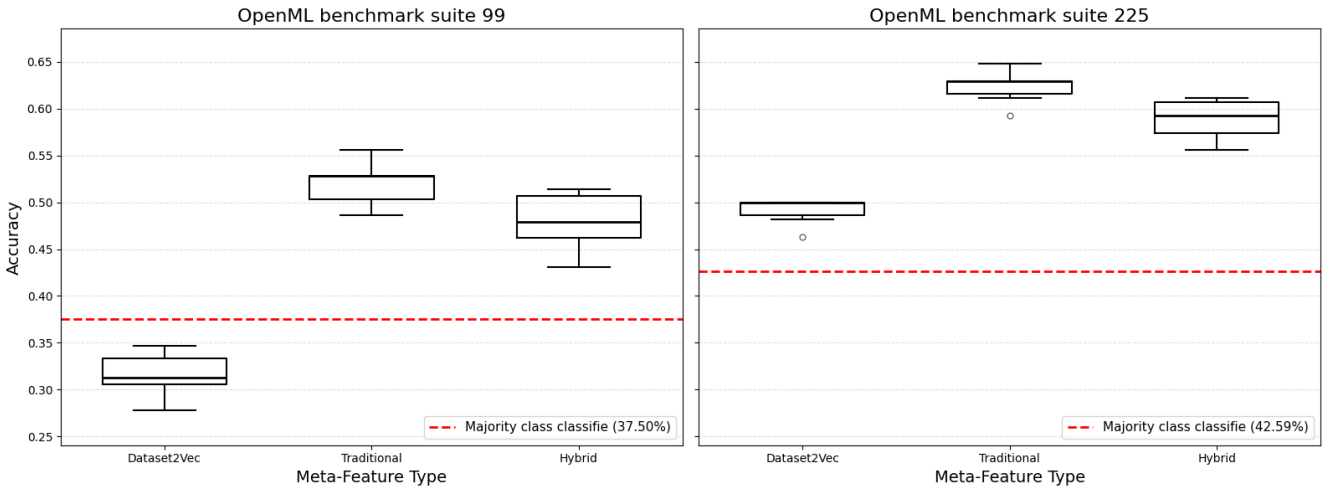


Figure 4: Distribution of meta-classifier performance across all base learners for the benchmark suites OpenML-CC18 (id=99) and OpenML100-friendly (id=225). The red dashed line indicates the baseline (majority class classifier). Results are shown for Dataset2Vec, traditional, and hybrid meta-feature representations.

Meta-Feature Type	**OpenML-CC18 (id=99)**	**OpenML100-friendly (id=225)**
	Meta-classifier accuracy	**Meta-classifier accuracy**
Traditional	0.528 ± 0.023	0.630 ± 0.017
Hybrid	0.479 ± 0.029	0.593 ± 0.021
Dataset2Vec	0.312 ± 0.022	0.500 ± 0.013
Majority-class baseline	0.375 ± 0.000	0.444 ± 0.000

Table 4: Meta-classifier performance for algorithm selection across the OpenML-CC18 and OpenML100-friendly benchmark suites. Reported values correspond to median accuracy with standard deviation over all base learners for traditional, Dataset2Vec, and hybrid meta-feature representations. The majority class classifier is included as a baseline.

Benchmark suite	**Feature set**	**MAE**	**MSE**	R^2
OpenML-CC18 (id=99)	Dataset2Vec	0.1316 ± 0.0007	0.0326 ± 0.0002	-0.222 ± 0.008
	Traditional	0.0720 ± 0.0006	0.0131 ± 0.0001	0.511 ± 0.005
	Hybrid	0.0729 ± 0.0006	0.0133 ± 0.0001	0.504 ± 0.005
OpenML100-friendly (id=225)	Dataset2Vec	0.1140 ± 0.0006	0.0242 ± 0.0003	0.140 ± 0.010
	Traditional	0.0562 ± 0.0006	0.0087 ± 0.0002	0.691 ± 0.006
	Hybrid	0.0551 ± 0.0003	0.0079 ± 0.0001	0.719 ± 0.005

Table 5: Meta-regressor performance across OpenML benchmark suites. Results are reported as mean $\pm$ standard deviation over repeated leave-one-out evaluations. Lower MAE and MSE indicate better performance, while a higher R^2 is preferable.

5.1.3 Comparison of Meta-classifier and Meta-regressor

In this section, we evaluate the meta-regressor in terms of algorithm-selection accuracy by converting its continuous performance predictions into discrete algorithm choices.

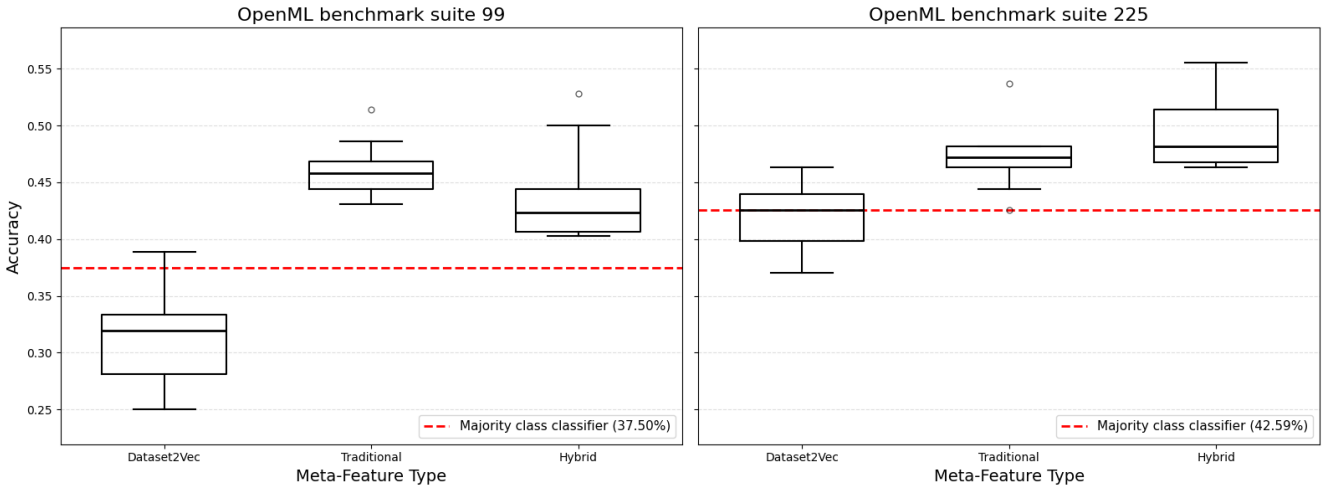


Figure 5: Distribution of meta-regressor performance across all base learners for the benchmark suites OpenML-CC18 (id=99) and OpenML100-friendly (id=225). For each dataset, the base learner with the highest predicted performance is selected. The red dashed line indicates the majority class classifier as the baseline. Results are shown for Dataset2Vec, traditional, and hybrid meta-feature representations.

Meta-Feature Type	**OpenML-CC18 (id=99)**	**OpenML100-friendly (id=225)**
	Meta-regressor accuracy	**Meta-regressor accuracy**
Traditional	0.461 ± 0.023	0.472 ± 0.028
Hybrid	0.439 ± 0.041	0.494 ± 0.031
Dataset2Vec	0.315 ± 0.040	0.422 ± 0.030
Majority-class baseline	0.375 ± 0.000	0.444 ± 0.000

Table 6: Meta-regressor performance for algorithm selection across the OpenML-CC18 and OpenML100-friendly benchmark suites. Reported values correspond to median accuracy with standard deviation for traditional, Dataset2Vec, and hybrid meta-feature representations. The majority class classifier is included as a baseline.

Figure 4 and Table 4 show that the meta-classifier consistently achieves higher algorithm-selection accuracy than the meta-regressor across both benchmark suites. For OpenML-CC18, traditional and hybrid meta-features perform well under direct meta-classification, while Dataset2Vec remains below the majority-class baseline. In contrast, the corresponding meta-regressor results in Figure 5 and Table 6 exhibit uniformly lower median accuracy and higher variance, despite preserving the same relative ranking of meta-feature representations. A similar pattern is observed for OpenML100-friendly

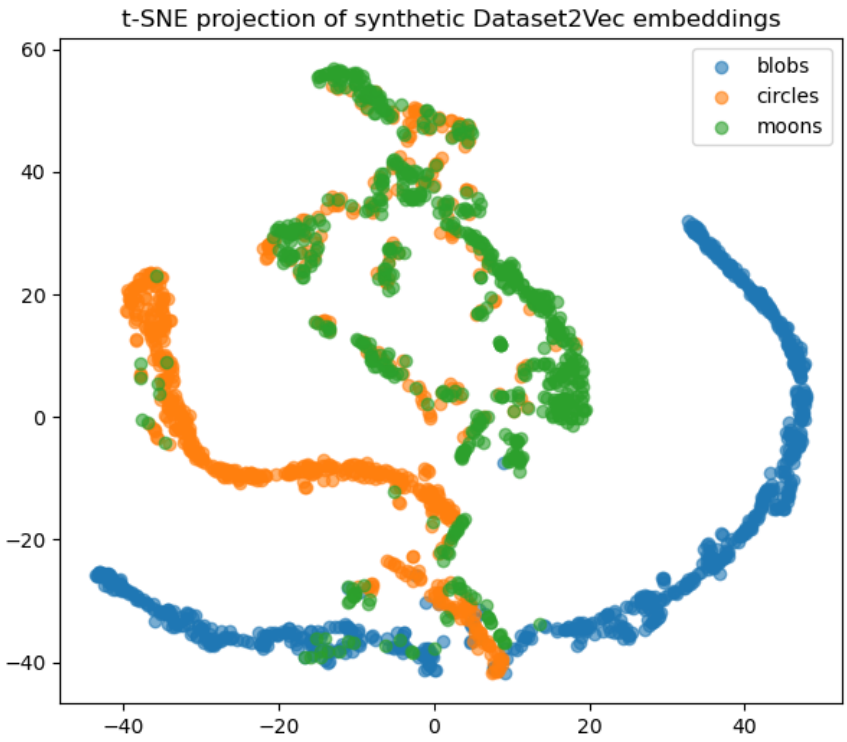


Figure 6: t-SNE projection of Dataset2Vec embeddings for synthetic toy datasets. Each point represents a dataset generated from one of three underlying geometric families, coloured by dataset type. The visualisation illustrates how Dataset2Vec representations organise datasets with similar geometric properties in the learned embedding space.

(id=225), where the meta-classifier substantially outperforms the meta-regressor, particularly for traditional meta-features. Overall, these results indicate that directly learning the algorithm-selection decision via meta-classification is more effective than regression-based selection.

5.2 Dataset2Vec representation analysis

Figure 6 shows that the Dataset2Vec embeddings capture a decent degree of structure among the synthetic datasets. Blob datasets are largely separated from the other groups, while circles and moons exhibit partial overlap, which is expected given their similar geometric characteristics. This qualitative pattern is consistent with the cosine distance statistics in Table 7. Intra-class distances are lowest for circles and moons, but the inter-class distance between these two groups is also small (0.0046 ± 0.0069), indicating limited but coherent separation. Blobs show larger intra-class variability and greater separation from the other dataset types, reflecting their more heterogeneous structure. Overall, these results suggest that Dataset2Vec provides some dataset structure, capturing broad similarities and differences between dataset types, while not enforcing perfectly distinct clusters.

Figure 7 shows t-SNE projections of synthetic datasets represented using hand-crafted information-theoretic and statistical PyMFE meta-features, which serve as the traditional baselines for comparison. In both cases, dataset types are more clearly separated than under Dataset2Vec, with visibly distinct clusters emerging in the embedding space.

Distance type	Groups	Cosine similarity
Intra-class	Blobs	0.0240 ± 0.0540
	Circles	0.0047 ± 0.0068
	Moons	0.0036 ± 0.0058
Inter-class	Blobs vs Circles	0.0249 ± 0.0248
	Blobs vs Moons	0.0291 ± 0.0232
	Circles vs Moons	0.0046 ± 0.0069

Table 7: Intra- and inter-class cosine distance statistics for Dataset2Vec representations of synthetic toy datasets, computed in the t-SNE projection space. Lower intra-class distances indicate tighter clustering of datasets with the same geometric structure, while higher inter-class distances indicate stronger separation between different dataset types.

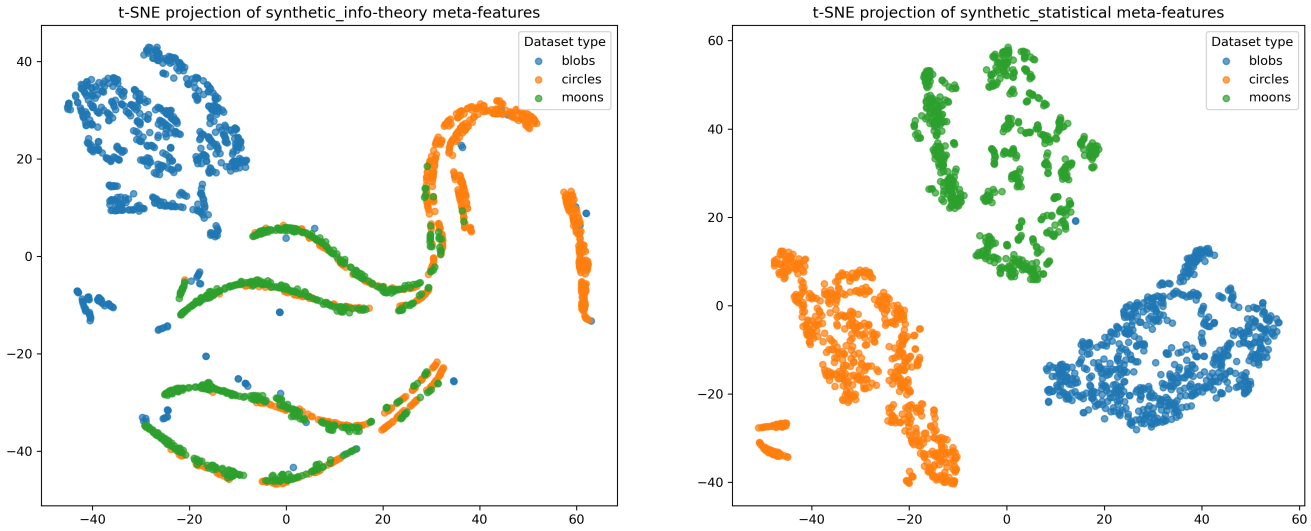


Figure 7: t-SNE projections of PyMFE information-theoretic and statistical meta-feature representations for synthetic datasets. Each point corresponds to a dataset, coloured by dataset type. This visualisation illustrates how traditional meta-features extracted using PyMFE organise datasets with similar geometric properties in the learned embedding space.

Representation	Avg. intra-class distance	Avg. inter-class distance	Inter/Intra ratio
Dataset2Vec	0.0108	0.0195	1.81
Information-theoretic	0.0311	0.1754	5.65
Statistical	0.1032	0.3685	3.57

Table 8: t-SNE-based separability comparison between Dataset2Vec and PyMFE meta-feature representations on synthetic datasets. Lower intra-class distances indicate tighter clustering of datasets with similar structure, while higher inter-class distances and inter/intra ratios indicate stronger separation between different dataset types.

Table 8 quantifies this difference in separability. Information-theoretic meta-features achieve a significantly higher average inter-class distance and the largest inter/intra ratio (5.65), indicating strong global separation between dataset types despite less compact intra-class structure. Statistical meta-features produce both large inter-class distances and visually compact clusters (Figure 7), resulting in a similarly strong inter/intra ratio (3.57).

Taken together, Figures 6 and 7, with Tables 7 and 8, show that while Dataset2Vec learns a partial representation of dataset structure, traditional PyMFE meta-features provide more discriminative embeddings on these synthetic benchmarks. This contrast highlights the role of traditional meta-features as strong baselines when assessing the expressiveness of learned dataset representations.

5.3 Algorithm subsets

In this section, we shifted the paradigm of the meta-learning task of algorithm selection by changing the set of base learners that the meta-model chooses from as defined in Section 4.7.

5.3.1 Algorithm structure

This section examines meta-model performance for algorithm selection when base learners are grouped according to their foundational methodological structure.

Figure 8 shows that across all structurally defined algorithm subsets, none of the meta-feature representations outperform the majority class classifier on either benchmark suite. Dataset2Vec meta-features consistently perform the worst, while traditional and hybrid meta-features yield similar performance, remaining close to but below the baseline. In the tree-based and ensemble subsets, high majority-class accuracies reflect severe class imbalance, with Random Forest dominating performance and limiting the meta-models' ability to learn when Decision Tree or XGBoost should be selected. Although meta-model performance is slightly higher in benchmark suite 225, this is likely due to larger performance gaps between algorithms, which provide a clearer discriminative signal despite increased imbalance. In the non-ensemble subset, the more balanced distribution of winners across tree-based, kernel-based, and linear methods results in a lower baseline; however, this increased algorithmic diversity still does not translate into meta-models surpassing the majority-class classifier, indicating that structural diversity alone does not improve algorithm selection performance.

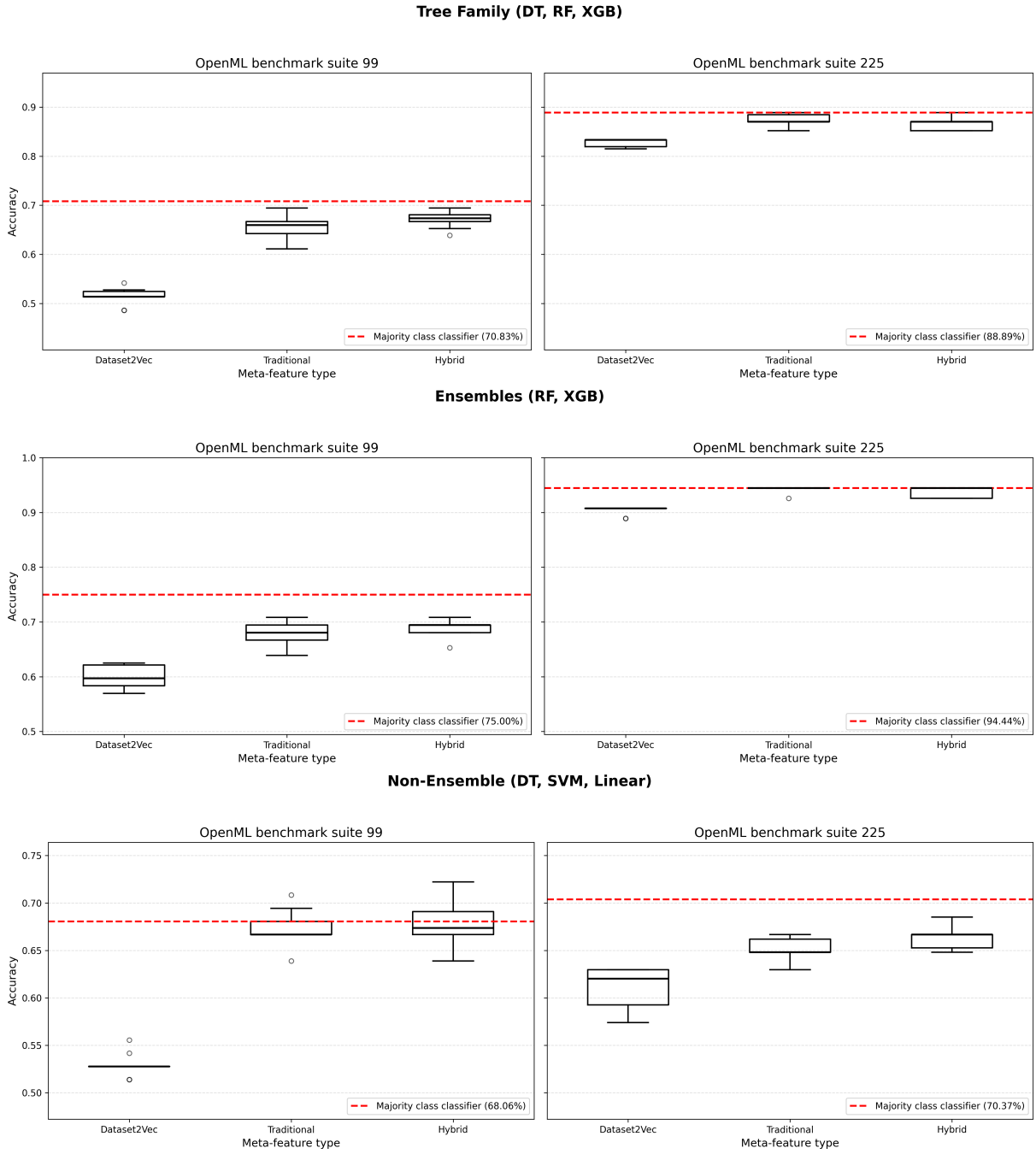


Figure 8: Meta-classifier performance across structure-based algorithm subsets. Boxplots show accuracy distributions for Dataset2Vec, traditional, and hybrid meta-feature representations on OpenML benchmark suites 99 and 225. Dashed red lines indicate the majority class classifier within each subset.

5.3.2 Empirical performance distribution

This section examines meta-model performance for algorithm selection when base learners are grouped according to their empirical algorithm performance across the benchmark suites.

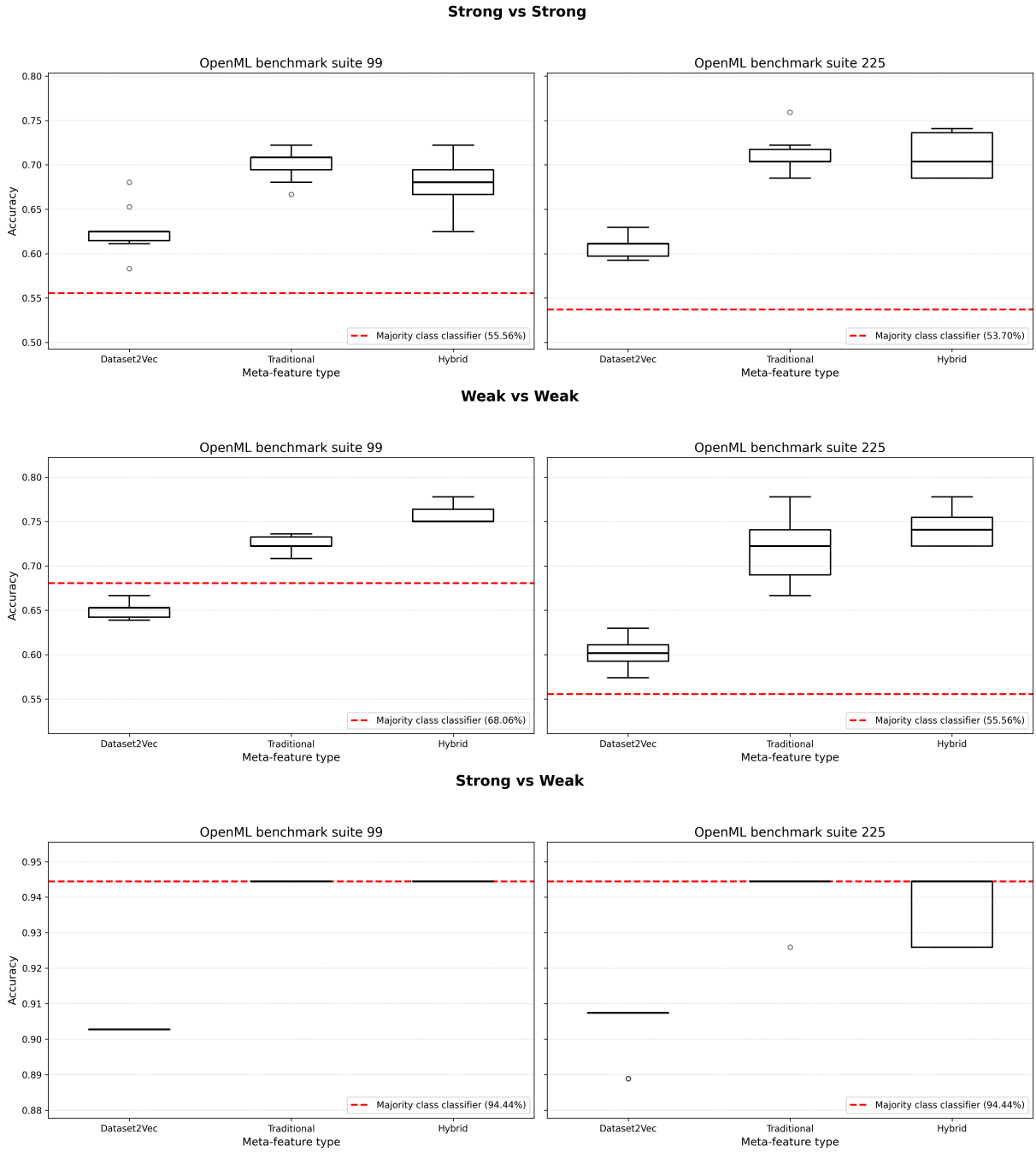


Figure 9: Meta-classifier performance across performance-based algorithm subsets. Boxplots show accuracy distributions for Dataset2Vec, traditional, and hybrid meta-feature representations on OpenML benchmark suites 99 and 225. Dashed red lines indicate the majority class classifier within each subset.

Figure 9 demonstrates that meta-model performance for algorithm selection strongly depends on the competitive balance between candidate base learners. When selecting between competitive algorithm pairs (strong vs. strong and weak vs. weak), all meta-feature types exceed or closely track the majority-class baseline. Traditional and hybrid meta-features benefit most under this condition, while Dataset2Vec exhibits weaker but still positive performance. In contrast, when selecting between non-competitive algorithm pairs, none of the meta-feature types outperform the baseline, indicating that the meta-model cannot improve upon the dominance of a single algorithm. Across all settings, traditional and hybrid meta-features consistently outperform Dataset2Vec meta-features. Overall, these results show that algorithm selection becomes effective primarily under competitive conditions, while dominance effects limit performance regardless of meta-feature representation.

6 Discussion and future work

This section interprets the experimental findings in relation to the research questions, as well as discussing limitations of the experiment and avenues for future research.

6.1 Research question 1 – How effective are deep learning extracted meta-features compared to traditional meta-features for algorithm selection in classification tasks?

Addressing the main research question — traditional meta-features outperform deep learning based extracted meta-features, using Dataset2Vec, consistently for algorithm selection. Traditional meta-features perform comparably to the hybrid meta-features, suggesting that Dataset2Vec does not add a substantial complementary signal to help the meta-model for algorithm selection. This is supported by consistent trends in the meta-classifier and meta-regressor, where Dataset2Vec not only underperforms the other meta-feature types but also fails to consistently exceed the baseline algorithm selection. The overall better meta-classifier performance compared to the meta-regressor suggests that algorithm selection is more effectively modelled as a classification task.

6.2 Research question 2 – To what extent do Dataset2Vec representations capture underlying structure relevant for algorithm selection

Dataset2Vec captures partial dataset structure, but not necessarily structure relevant for algorithm selection. The t-SNE projection on synthetic datasets indicates that Dataset2Vec creates representations that distinguish between some dataset types (e.g., blobs), while failing to separate geometrically similar categories (e.g., circles and moons). Traditional meta-features extracted using PyMFE achieve stronger separation, suggesting that Dataset2Vec representations are less informative for algorithm selection.

6.3 Research question 3 – How does varying base learners affect the performance of meta-models trained on different types of meta-features

Performance of meta-models for algorithm selection varies significantly when the algorithm paradigm is changed — they exhibit reduced performance under algorithm foundation comparisons and improved performance under competitive performance comparisons. When comparing tree family, ensemble, and non-ensemble methods, all sets of meta-features underperform the baseline, at best performing close to the baseline. This indicates that meta-features in general have a limited ability to distinguish between different methodologically based comparisons. In balanced settings, all meta-feature types outperform the baseline. It suggests that class imbalance and algorithm dominance limit meta-learning effectiveness.

6.4 Remarks

Dataset2Vec demonstrates effectiveness in hyper-parameter optimisation [JSTG21], but shows limited effectiveness in algorithm selection, highlighting fundamental differences between the two meta-learning tasks.

In hyper-parameter optimisation, Dataset2Vec serves as an auxiliary signal to identify similar datasets and uses their hyper-parameter configurations as a warm start for further optimisation. In algorithm selection, Dataset2Vec representations serve as the primary discriminative basis for prediction, with no subsequent optimisation to correct for imperfect meta-features. This task requires more precision, as the smaller search space leaves no space for approximate guidance.

At its core, Dataset2Vec uses the auxiliary task of dataset similarity to construct the vector representations. This approach is well-suited for hyper-parameter optimisation, where similar datasets in a fixed algorithm space generally share the same optimal hyper-parameter configurations. In contrast, algorithm selection cannot rely on this assumption: similar datasets in the Dataset2Vec embedding space can require different optimal algorithms. Algorithm selection can depend on specific discriminative characteristics that do not correlate with dataset similarity. Dataset2Vec compressing a whole dataset into a vector representation aggregates these characteristics. Traditional meta-features, in comparison, explicitly preserve those characteristics. This mismatch between the auxiliary task of Dataset2Vec and the algorithm selection suggests that its limited performance may stem from objective misalignment rather than an inherent limitation of deep learning–based meta-features. Fine-tuning Dataset2Vec directly on an algorithm selection objective would therefore be a necessary next step to assess its true potential in this setting.

6.5 Limitations

Several limitations contextualise these findings. First, the evaluation focuses on five algorithms (SVM, Random Forest, Decision Tree, XGBoost, Linear Models) and tabular classification tasks; findings may not generalise to larger algorithm portfolios, modern approaches (neural networks, AutoML), or other problem domains (computer vision, NLP, time-series). Second, this study uses the pre-trained Dataset2Vec model without retraining or fine-tuning on the algorithm selection task, which may limit the effectiveness of Dataset2Vec representations for discriminative algorithm selection. Alternative

training procedures or pre-training data may yield different results. Third, the 126 datasets across two benchmark suites (OpenML-CC18 and OpenML100-friendly) represent a relatively small number of meta-learning instances, limiting meta-model complexity and reliability. Despite these limitations, consistent patterns across experimental conditions and systematic evaluation across algorithm subsets support the findings' validity.

6.6 Future work

The findings of this thesis suggest avenues for future research:

Improved deep learning approach. Implementing deep learning meta-feature extraction trained directly on an algorithm selection objective instead of dataset similarity would test whether the limitation is Dataset2Vec-specific or fundamental to deep learning. For example, representations could be trained to predict base learner performance rankings, allowing the meta-model to learn algorithm-discriminative signals directly.

Feature importance analysis. Systematic analysis of which meta-features most strongly predict algorithm performance would reveal essential discriminative properties. Feature importance analysis, ablation studies, and correlation analysis would identify what traditional meta-features capture that Dataset2Vec misses.

Discriminative feature engineering. Informed by importance analysis, developing novel meta-features targeting algorithm-discriminative properties would improve both approaches. This includes identifying new properties, creating composite features, and investigating domain-specific characterisations.

7 Conclusion

In this thesis, we built a tool that leverages OpenML to construct a meta-learning system for algorithm selection and use it to compare models trained on traditional, Dataset2Vec, and a combination of both types of meta-features. We found that Dataset2Vec meta-features underperform traditional approaches across both benchmark suites, performing close to the baseline. Dataset2Vec captures partial dataset structure, as evidenced by weak but coherent clustering in t-SNE projections of synthetic data, but shows weaker discriminative power than traditional meta-features and proves insufficient for algorithm selection. Furthermore, while traditional meta-features generally perform best, the effectiveness ranking varies across algorithm subsets. Algorithm subset analysis demonstrated that meta-learning effectiveness depends critically on both algorithmic relationships and competition patterns. Meta-features excel at distinguishing between diverse algorithmic paradigms but struggle with subtle variations within algorithm families, and prove most valuable in balanced competition scenarios while providing minimal value when one algorithm clearly dominates. We conclude that meta-feature effectiveness depends not on descriptive power but on alignment with the specific meta-learning task. Task-specific relevance must be prioritised over general dataset characterisation in meta-feature design.

References

[Aha92] David W Aha. Generalizing from case studies: A case study. In *Proceedings of the Ninth International Workshop on Machine Learning*, 1992.

[ASR+20] Edesio Alcobaça, Felipe Siqueira, Adriano Rivolli, Luís P. F. Garcia, Jefferson T. Oliva, and André C. P. L. F. de Carvalho. MFE: Towards reproducible meta-feature extraction. *Journal of Machine Learning Research*, 21(111), 2020.

[BCD+25] Bernd Bischl, Giuseppe Casalicchio, Taniya Das, Matthias Feurer, Sebastian Fischer, Pieter Gijsbers, Subhaditya Mukherjee, Andreas C Müller, László Németh, Luis Oala, Lennart Purucker, Sahithya Ravi, Jan N van Rijn, Prabhant Singh, Joaquin Vanschoren, Jos van der Velde, and Marcel Wever. OpenML: Insights from 10 years and more than a thousand papers. *Patterns*, 6(7):101317, 2025.

[BCF+21] Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan N van Rijn, and Joaquin Vanschoren. OpenML benchmarking suites. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021*, 2021.

[BGC00] Hilan Bensusan and Christophe G. Giraud-Carrier. Discovering Task Neighbourhoods through Landmark learning performances. In *Principles of Data Mining and Knowledge Discovery, 4th European Conference (PKDD 2000)*. Springer, 2000.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, 1 edition, 2006.

[BK01] Hilan Bensusan and Alexandros Kalousis. Estimating the predictive accuracy of a classifier. In *European Conference on Machine Learning (ECML)*, volume 2167 of *Lecture Notes in Computer Science*, pages 25–36. Springer, 2001.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[BvRSV22] Pavel Brazdil, Jan N. van Rijn, Carlos Soares, and Joaquin Vanschoren. *Metalearning: Applications to Automated Machine Learning and Data Mining*. Cognitive Technologies. Springer Cham, 2 edition, 2022.

[CG16] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[CGL23] Nathan Carvalho, André Gonçalves, and Ana Lorena. Collecting meta-data from the OpenML public repository. In *Anais do XX Encontro Nacional de Inteligência Artificial e Computacional*, pages 610–624, Porto Alegre, RS, Brasil, 2023. SBC.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[Fuj25] Keisuke Fujii. *Machine Learning in Sports: Open Approach for Next Play Analytics*. Springer Singapore, 1 edition, 2025.

[GC08] Christophe Giraud-Carrier. Metalearning: A tutorial, 2008. Obtained from [https://www.icmla-conference.org/icmla08/slides2.pdf].

[GLH15] Salvador García, Julián Luengo, and Francisco Herrera. *Data Preprocessing in Data Mining*. Intelligent Systems Reference Library. Springer Cham, 1 edition, 2015.

[HK01] Melanie Hilario and Alexandros Kalousis. Fusion of meta-knowledge and meta-data for case-based model selection. In *European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, volume 2168 of *Lecture Notes in Computer Science*, pages 180–191. Springer, 2001.

[JSTG21] Hadi S. Jomaa, Lars Schmidt-Thieme, and Josif Grabocka. Dataset2vec: Learning dataset meta-features. *Data Mining and Knowledge Discovery*, 35(3):964–985, 2021.

[Koh95] Ron Kohavi. A study of crossvalidation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1137–1143, 1995.

[KTK00] Christian Köpf, Charles Taylor, and Jörg Keller. Meta-analysis: From data characterization for meta-learning to meta-regression. In Pavel Brazdil and Amílcar Jorge, editors, *Proceedings of the PKDD 2000 Workshop on Data Mining, Decision Support, Meta-Learning and ILP: Forum for Practical Problem Presentation and Prospective Solutions*, pages 15–26, 2000.

[LBNA+03] Kevin Leyton-Brown, Eugene Nudelman, Galen Andrew, Jim McFadden, and Yoav Shoham. A portfolio approach to algorithm selection. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, page 1542. Morgan Kaufmann, 2003.

[MST94] Donald Michie, David J. Spiegelhalter, and Charles C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

[PBGC00] Bernhard Pfahringer, Hilan Bensusan, and Christophe Giraud-Carrier. Meta-learning by landmarking various learning algorithms. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pages 743–750, 2000.

[PKP22] Someswari Perla, Naga Nimesh K, and Srinidhi Potta. Implementation of autonomous cars using machine learning. In *2022 International Conference on Edge Computing and Applications (ICECAA)*, pages 1444–1451, 2022.

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[RGS+19] Adriano Rivolli, Luís P. F. Garcia, Carlos Soares, Joaquin Vanschoren, and André C. P. L. F. de Carvalho. Characterizing classification datasets: a study of meta-features for meta-learning, 2019.

[Ric76] John R Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

[Smi08] Kate Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv.*, 41(1):6:1–6:25, 2008.

[Sto74] Mervyn Stone. Crossvalidatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B*, 36(2):111–147, 1974.

[Van18] Joaquin Vanschoren. Meta-learning: A survey, 2018.

[VvRBT13] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luís Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2013.

[ZKR+17] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pages 3391–3401, 2017.

[ZSW23] B. Zhang, H. Shi, and H. Wang. Machine learning and ai in cancer prognosis, prediction, and treatment selection: A critical approach. *Journal of Multidisciplinary Healthcare*, 16:1779–1791, 2023. Published 2023.