A Distance-Weighted Class-Homogeneous Neighbourhood Ratio for Algorithm Selection

Haofei Chen CHENHF-MS@HOTMAIL.COM
Ya Liu LIUYA2360@GMAIL.COM
Japnit Kaur Ahuja JAPNIT.AHUJA@GMAIL.COM
National Junior College, 37 Hillcrest Rd, Singapore 288913
Daren Ler DLER@COMP.NUS.EDU.SG
National University of Singapore, 21 Lower Kent Ridge Rd, Singapore 119077

Editors: Sinno Jialin Pan and Masashi Sugiyama

Abstract

In this paper, we introduce a new form of meta-feature that is based on a distance-weighted class-homogeneous neighbourhood ratio to facilitate algorithm selection. We show that these new meta-features, while exhibiting a cost advantage, achieve a comparable, and in some cases, higher performance than conventional meta-features. These results were obtained via experiments conducted over artificial datasets and real-world datasets from the UCI repository. We further redefine the algorithm selection problem by advocating that accuracy should be calculated based on the assumption that the population of datasets is uniformly distributed. Finally, in this paper, we provide a new perspective on landmarkers, such that a landmarker corresponds to a tuple (algorithm, metric), and propose the idea of a new family of meta-features.

Keywords: Meta-learning, Algorithm Selection, Meta-features, Landmarking, AutoML

1. Introduction

According to the No Free Lunch Theorems (Wolpert, 1996b,a), there is no single algorithm that uniformly outperforms every other algorithm over all problems, thus necessitating algorithm selection. However, given the plethora of new algorithms available, the task of choosing an algorithm has become complicated and time-consuming. Consequently, the need for algorithm selection mechanisms that do not require human intervention has increased, shifting the attention to Automatic Machine Learning (AutoML). As a result, several automated versions of popular machine learning platforms have been implemented, including Auto-Weka (Thornton et al., 2012) and Hyperopt-sklearn (Komer et al., 2014).

Since the seminal work by Rice (1976), significant research has been done on algorithm selection via meta-learning, which utilises various meta-features and learning algorithms (Aha, 1992; Brazdil et al., 2008; Ali and Smith, 2006; Lee and Giraud-Carrier, 2013; Brazdil and Giraud-Carrier, 2018). Moreover, it has inspired the expansion of meta-learning to solve various types of problems (Smith-Miles, 2009; Vilalta and Drissi, 2002; Kalousis and Hilario, 2001; Ali and Smith, 2006).

In this paper, we propose and experiment on a new form of meta-feature that is based on a distance-weighted class-homogeneous neighbourhood ratio. In our experiments, we

compare the proposed meta-features against more conventional meta-features based on both the typical notion of accuracy, and on one that assumes that the population of all datasets (at least in terms of algorithm superiority) is uniform. Essentially, we assume that most real-world repositories would not be drawn *i.i.d.* from the population of all classification problems, and as such, the datasets from such repositories would probably not represent the unknown distribution governing the population of all possible classification problems.

Our results show that the proposed meta-features, while being more computationally efficient, are comparable to conventional meta-features, especially when trained on imbalanced data.

This paper summarises related work that has been done in the field of meta-learning in Section 2. In Section 3, we propose a new perspective on landmarkers, and suggest new meta-features based on a distance-weighted class-homogeneous neighbourhood ratio. This is then followed by Section 4, which describes the experiments conducted over artificial and real-world datasets to compare the performance of our proposed meta-features to that of conventionally used meta-features. Then, in Section 5, we analyse the results obtained from the experiment, before concluding with a summary of the findings and future work.

2. Related Work

Adapted from Rice (1976), Figure 1 describes the prototypical algorithm selection framework. A base-level dataset D is a sample classification problem drawn (assumedly *i.i.d.*) from the population that governs classification problems in the problem space $\mathcal{P}$. Each base-level algorithm a in the algorithm space $\mathcal{A}$ is trained over D to produce a base-level model, $a(D) = m_a$. When the model is tested on unlabelled instances (i.e., from the test set), the performance Π of each model can be computed with some evaluation method E, i.e., $E(m_a) = \Pi_{m_a}$. A meta-label generator T takes each model's performance as an input, and computes the meta-label of the dataset, $T(\{\Pi_{m_a} | a \in \mathcal{A}\}) = Y_D$. The meta-features X_D are generated using a feature-extraction function G, i.e., $G(D) = X_D$.

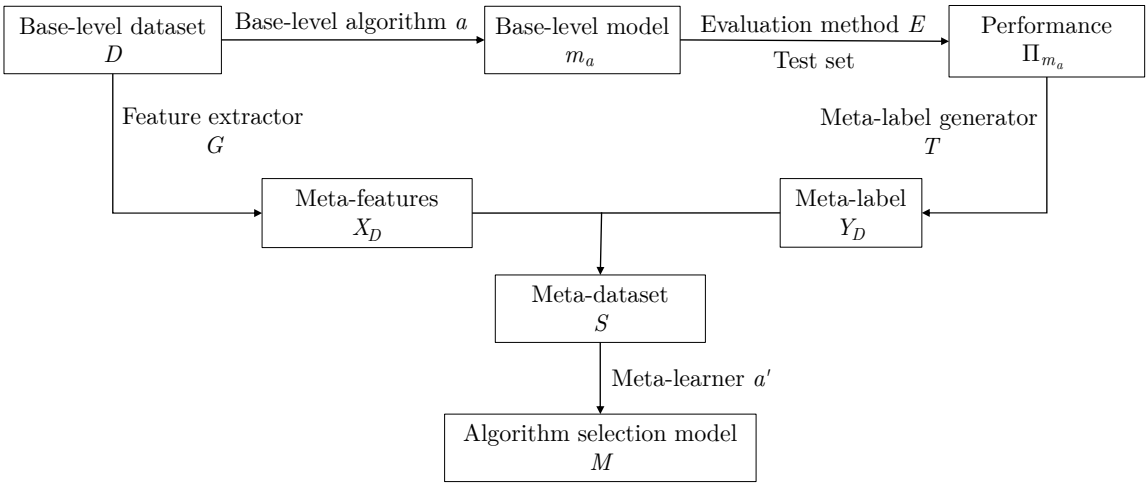

Figure 1: The algorithm selection framework.

The meta-features X_D and meta-labels Y_D of a base-level dataset D together form one instance in the meta-dataset $S = \{\langle X_D, Y_D \rangle | D \in \mathcal{P}\}$. The algorithm selection model M is produced by training a meta-learner a' over the meta-dataset S, $a'(S) = M$. The algorithm selection model M takes a new base-level dataset D' as input and predicts a suitable algorithm $a^\star$ for it, such that the base-level model $a^\star(D') = m_{a^\star}$ has the highest performance under the specified evaluation method E. That is, $a^\star = \arg\max_{a \in \mathcal{A}} \Pi_{m_a}$.

The chosen meta-features X_D should capture the complexity and other characteristics of the base-level dataset D (Smith-Miles, 2009). The Statlog Project (Michie et al., 1994) categorised 16 dataset characteristics into 3 groups: simple measures, statistical measures and information theory measures. Further research contributed to the domain of meta-features with the addition of landmarkers (Pfahringer et al., 2000) and meta-features based on the structural properties of a model (Peng et al., 2002). A more comprehensive review of meta-features can be found in a recent survey by Vanschoren (2018).

In a recent review (Lorena et al., 2019), complexity measures based on different structures were discussed. However, in that work different types of complexity were not definitively categorised. Building on the factors of complexity for classification problems given by Ho and Basu (2002), we define the following specific categories of complexity:

1. Complexity of decision boundary: The decision boundary estimates the difficulty in separating the instances of different classes and assigning a class to a new instance.
2. Imbalance: Proportion of instances in each class.
3. Spatial distribution of the data
    - Class-wise coverage: the distribution of the instances within each class.
    - Overall coverage of the space: the distribution of the instances in the whole dataset.

While a significant proportion of the work done on algorithm selection has been focused on meta-features, other variations in the prototypical framework include changes in the meta-learner and the meta-label.

Any traditional algorithm used in base-level machine learning can be used as a meta-learner. This includes decision trees (Pfahringer et al., 2000), support vector machines (Kim et al., 2017) and neural networks (Mishra et al., 2017). Another conventional meta-learner, which is also used in this paper, is k-Nearest Neighbours (Brazdil et al., 2008; Pfahringer et al., 2000).

There are three different ways of generating meta-labels. The first returns a single superior algorithm $a^\star$ (such that $a^\star = \arg\max_{a \in \mathcal{A}} \Pi_{m_a}$). The second selects a group of algorithms, including the most superior one and others whose performance are not significantly different. The third approach ranks all the algorithms (Brazdil and Soares, 2000) via ranking-based systems such as predictive clustering trees (Todorovski et al., 2002). Other methods introduce measures that combine accuracy and time (van Rijn et al., 2015; Soares and Brazdil, 2000).

3. A New Perspective on Landmarkers

3.1. Prototypical Landmarkers and a New Perspective

A prototypical landmarker has been described as a computationally cheap and efficient algorithm a. A model m_a is generated by training a using dataset D. Typically, the performance Π_{m_a} serves as the meta-feature, with the evaluation method E corresponding to prediction accuracy (Pfahringer et al., 2000). Such meta-features derived from the landmarkers provide a more direct characterisation of the dataset, as compared to conventional meta-features, which tend to measure very specific characteristics that may not be relevant to the chosen set of base-level algorithms.

The two criteria initially proposed for selecting an algorithm to be used as a landmarker are efficiency and bias diversity (Pfahringer et al., 2000). Efficiency is a criterion since we wish to reduce computational cost, while the bias diversity criterion is based on the desire to provide a wider spectrum of dataset characteristics.

This definition of a landmarker assumes the use of a static performance metric Π_{m_a} to serve as a meta-feature. We extend the definition of the evaluation method E and propose a new definition for landmarkers: a landmarker $(a,\ E)$ consists of *any* single algorithm a that is applicable to the dataset D being characterised, and *any* metric E that is measurable on the model m_a, where $a(D) = m_a$. There are many potential metrics that may be used as E, including, but not limited to: (i) the conventionally used performance metric Π_{m_a}, e.g. accuracy of the model m_a, and (ii) the characteristics of m_a itself, e.g. number of leaves and nodes of m_a, when m_a is a decision tree model (Peng et al., 2002).

In this paper, we introduce a new metric: a distance-weighted class-homogeneous neighbourhood ratio, to facilitate algorithm selection.

3.2. The Distance-Weighted Class-Homogeneous Neighbourhood Ratio

Consider a dataset D with n labelled instances, each having an m-dimensional feature vector $\mathbf{x}$. That is, $D = \{\langle \mathbf{x}_1, y_1\rangle, \langle \mathbf{x}_2, y_2\rangle, \ldots, \langle \mathbf{x}_n, y_n\rangle\}$, where $\mathbf{x}_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\}$, and $y_1, y_2, \ldots, y_n$ are the labels. The distance-weighted class-homogeneous neighbourhood ratio β, of instance $\langle \mathbf{x}_i,\ y_i\rangle$, is defined as:

$$\beta_{\langle \mathbf{x}_i, y_i\rangle} = \frac{\sum_{\langle \mathbf{x}_j,\ y_j\rangle \in NN\left(\langle \mathbf{x}_i,\ y_i\rangle,\ u_{y_i}\right) | y_i = y_j} invd\left(\mathbf{x}_i, \mathbf{x}_j\right)}{\sum_{\langle \mathbf{x}_j,\ y_j\rangle \in D \setminus \{\langle \mathbf{x}_i,\ y_i\rangle\}} invd\left(\mathbf{x}_i, \mathbf{x}_j\right)} \tag{1}$$

where

$$u_{y_i} = |\{\langle \mathbf{x}_j, y_j\rangle \in D | y_i = y_j\}| - 1 \tag{2}$$

$$NN\left(\langle \mathbf{x}_i,\ y_i\rangle, u_{y_i}\right) = \{\langle \mathbf{x}_j,\ y_j\rangle \in D \setminus \{\langle \mathbf{x}_i,\ y_i\rangle\} \,|\, \|\mathbf{x}_i - \mathbf{x}_j\| < d(\mathbf{x}_i, u_{y_i})\} \tag{3}$$

$$invd\left(\mathbf{x}_i,\ \mathbf{x}_j\right) = \frac{1}{1 + \|\mathbf{x}_i - \mathbf{x}_j\|} \tag{4}$$

and $d(\mathbf{x}_i, u_{y_i})$ is the Euclidean distance between the instance with feature vector $\mathbf{x}_i$ and the u_{y_i}-th nearest instance to it. In plain words, the β-value of an instance $\langle \mathbf{x}_i, y_i\rangle$ is defined as

the ratio of: (i) the sum of the inverse-distances (Equation 4) over instances with label y_i among the u_{y_i} nearest neighbours of $\langle \mathbf{x}_i, y_i \rangle$, where the value of u_{y_i} is the number of other instances in the dataset with label y_i, to (ii) the sum of the inverse-distances over all other instances in the dataset. Our proposed landmarker is thus $(a^\star, \beta)$, where $a^\star$ is an algorithm that, when trained over D, produces the target labelling function $f^\star$.

The β-value of the instances in a labelled dataset reveal the distribution of instances within a class (class-wise coverage) relative to the distribution in the whole dataset (overall coverage of the space). This can be illustrated in the following example.

Consider the simple classification problem shown in Figure 2. The feature space is a 2-dimensional Euclidean plane and there are only two distinct clusters, each containing instances of a different class. The diameter of the blue cluster, representing class 0, is d_1, and that of the orange cluster, representing class 1, is d_2. The separation between the two clusters is d_3. The figure illustrates high cohesion (compact clusters) and high separation ($d_3 \geq \max(d_1, d_2)$). Under our definition of u_{y_i} in Equation 2, the "nearest neighbours" (as produced by Equation 3) for each instance in the example, correspond to all the instances in the same cluster (since those all of instances share the same class). This maximises the numerator of Equation 1, and thus maximises the β-value of each instance. As the separation increases, i.e., d_3 tends to infinity, the proportion of weighted distances to instances of the other class in the denominator of Equation 1 becomes smaller, and the β-value of all instances tends to 1 (note: β can be equal to 1 in special cases where all the instances in a dataset are of the same class).

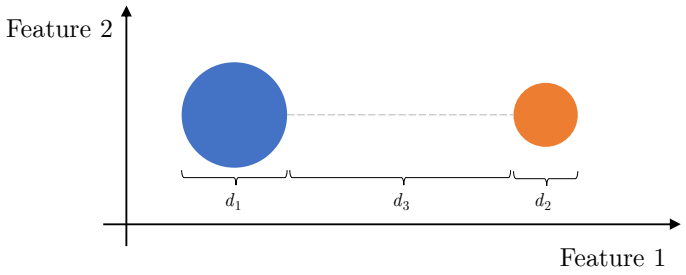

Figure 2: A simple classification problem.

Consider a slight variation of the above example, where an instance of class 1 (orange) is added to the cluster of class 0 (blue). One should expect that this newly added instance would be difficult to correctly classify. For this instance, all of its "nearest neighbours", as produced by Equation 3, are in the blue cluster – these are all of the opposite class as compared to the instance in question. The numerator in Equation 1 of that added instance is 0, as is its β-value.

We can see that the β-value of an instance considers both cluster cohesion and separation, and suggests the complexity of classification at instance level. The range of β is [0,1].

β is inspired from the silhouette score (Rousseeuw, 1987), s, such that:

$$s = \frac{p - q}{\max(p, q)} \tag{5}$$

where p is the mean distance between an instance and all other instances in the same cluster, and q is the mean distance between an instance and all other instances in the neighbouring cluster. s lies in the the interval [-1, 1].

The silhouette score measures how well an instance is positioned within its own cluster (cohesion) relative to other clusters (separation). Typically, the mean silhouette score (over all instances) is used to characterise a dataset, with higher values usually indicative of datasets containing distinct clusters. In a similar fashion, β measures how well-positioned an instance is in the dataset depending on its neighbourhood of instances with the same class. However, instead of using mean distances, as is the case with the silhouette score (Equation 5), β uses the sum of weighted distances to capture more information. Like the silhouette score, higher values of β signify a simpler classification problem.

β and the silhouette score have a comparable run-time complexity. The computational complexity of β is largely bounded by: (i) the calculation of Euclidean distances between each pairs of instances, and (ii) sorting these distances in order to find the "nearest neighbours" (Equation 3). More precisely, this complexity is bounded by $O(n^2(m + \log n))$. In comparison, the run-time complexity of the silhouette score is $O(n^2 m)$ (Vendramin et al., 2010). Essentially, the run-time complexity of β is slightly higher than that of the silhouette score.

One key difference between β and the silhouette score is the consideration of class information. Silhouette score is independent of class information and is based solely on construction of distinct spherical clusters, whereas β takes into account the class information.

To form meta-features from β-values, we can use a k-bin histogram to aggregate the β-value of all instances in a dataset D. This produces a k-dimensional vector, $\{bin_1, bin_2, \ldots, bin_k\}$, where the value of $bin_i, i \in \{1, ..., k\}$, is the proportion of instances in D that have β-value falling in the range of $[\frac{1}{k}(i-1), \frac{1}{k}i)$. Thus, this vector is independent of the size of the dataset. This method of utilising k-bin β-frequencies thus facilitates an adjustable granularity over the information captured – i.e., by increasing the number of bins, more information can be captured. We hypothesise that this method of filtering β-values into k bins of β-frequencies represents valuable information, which would be useful in characterising base-level datasets for the purposes of performing algorithm selection.

It should be noted that conventional meta-features do not typically include the silhouette score. Most complexity-based meta-features instead characterise datasets based on some general structure (Lorena et al., 2019). Since β captures information at the instance-level, we hypothesise that it may capture more information than the typical complexity-based meta-feature.

3.3. Preliminary Experiment on β-frequencies

To establish that β-frequencies capture the information that other complexity measures do, the following preliminary experimentation was conducted. We generated binary-classed base-level artificial datasets of different configurations in a 2-D Euclidean plane as shown in Figure 3, where each dataset $D = \{\langle \mathbf{x}_i = \{x_{i,1}, x_{i,2}\}, y_i \in \{0, 1\}\rangle\}$. The blue clusters signify

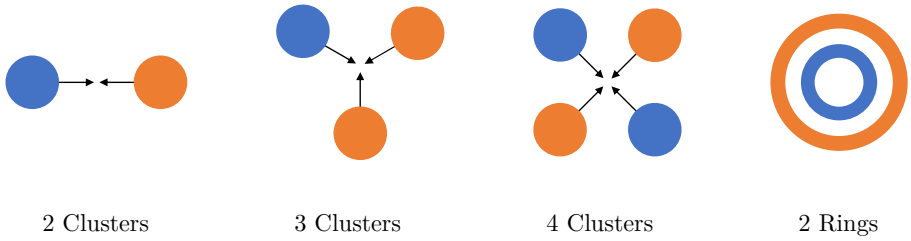

Figure 3: Configurations of artificial datasets.

class 0 and the orange clusters signify class 1. Each dataset had 1000 randomly generated instances. All the clusters initially had a separation of 290 units from the centroid, and in each iteration the separation was reduced by 10 units, causing the clusters to move towards the centroid and eventually merging together. We generated 20 datasets in each iteration, amounting to a total of 600 datasets for each configuration. The datasets generated had their mean silhouette score distributed in the range [-0.0008,0.9].

After trying various sizes for the k-bin β-frequencies, we found that there was not much improvement in the information captured for $k > 10$. Thus, we have used 10-bin β-frequencies. For each dataset, we calculated the R^2 value for the linear regression model, where the dependent variable is a complexity measure, and the independent variables corresponded to the 10-bin β-frequencies. The results are listed in Table 1.

The 10-bin β-frequencies capture much of the information that several feature-based complexity measures ($F1$, $F1v$, $F2$, $F3$ and $F4$) do. This is observed via the high R^2-values over most dataset configurations (i.e., all except 4-cluster datasets and some 2-ring datasets). Since β is based on the sum of weighted distances, it indirectly measures how discriminating the features in the base-level dataset are (β is higher for instances with neighbourhoods of mostly the same class – i.e., the feature values over instances with the same class vary less). However, it is only able to represent part of the information. In more complicated datasets, such as the 4-cluster and the 2-ring datasets, there is a huge overlap between the class-specific feature-value ranges due to their topology, and consequently, the β-frequencies are unable to capture the same information present in the feature-based complexity measures.

Expectedly, the 10-bin β-frequencies are able to capture the information in neighbourhood and network measures ($N1$, $N2$, $N3$, $N4$, $Density$, $ClsCoef$ and $Hubs$), as seen by the high-levels of correlation, which indicates that β-frequencies represent the density of same class in the neighbourhood of an instance.

The 10-bin β-frequencies also approximately capture the same information as the linearity measures ($L1$, $L2$, and $L3$). This is the case for all dataset configurations apart from the 4-cluster datasets. Since the 10-bin β-frequencies summarise how the data is distributed within a class relative to the whole dataset, it represents the complexity of the decision boundary. However, as the linearity measures do not explicitly measure the decision boundary, they inadequately capture the information corresponding to small changes in the β-frequencies, as seen in the complex case of the 4-cluster datasets.

Table 1: R^2 values of β-frequencies vs complexity measures.

Measure	**2 Clusters**	**3 Clusters**	**4 Clusters**	**2 Rings**
Maximum Fisher's discriminant ratio ($F1$)	0.9995	0.9996	0.4124	0.0075
Directional vector maximum Fisher's discriminant ratio ($F1v$)	0.9988	0.9969	0.4348	0.0819
Volume of overlapping region ($F2$)	0.9933	0.9948	0.5489	0.9958
Maximum individual feature efficiency ($F3$)	0.9982	0.9992	0.0074	0.9952
Collective feature efficiency ($F4$)	0.9993	0.9992	0.0098	0.9904
Faction of borderline points ($N1$)	0.9993	0.9980	0.9977	0.9947
Ratio of intra/extra class NN distance ($N2$)	0.9988	0.9959	0.9876	0.9983
Error rate of NN classifier ($N3$)	0.9976	0.9977	0.9959	0.9947
Non linearity of NN classifier ($N4$)	0.9947	0.9969	0.9927	0.9832
Fraction of hyperspheres covering data ($T1$)	0.9985	0.9981	0.9971	0.9965
Local set average cardinality (LSC)	0.9946	0.9992	0.9993	0.9956
Sum of the error distance by linear programming ($L1$)	0.9976	0.9986	0.6373	0.9630
Error rate of linear classifier ($L2$)	0.9989	0.9993	0.1961	0.9395
Non linearity of linear classifier ($L3$)	0.9945	0.9967	0.1917	0.8466
Average number of features per dimension ($T2$)	0.0000	0.0000	0.0000	0.0000
Average number of PCA dimensions per points ($T3$)				
Ratio of the PCA dimension to the original dimension ($T4$)				
Entropy of classes proportions ($C1$)				
Imbalance ratio ($C2$)				
Density	0.9978	0.9961	0.9984	0.9976
Clustering Coefficient ($ClsCoef$)	0.9730	0.9876	0.9857	0.9468
Hubs	0.5726	0.8621	0.4506	0.7202

Overall, the 10-bin β-frequencies have a high correlation with most complexity meta-features. However, it is noteworthy that β-frequencies have an advantage in efficiency. β-frequencies are computationally cheaper, requiring only $O(n^2(m+\log n))$ time, as compared to the more expensive complexity meta-features, such as $F1v$, which has complexity $O(m \cdot n \cdot n_c + m^3 \cdot n_c^2)$, where n_c is the number of classes in the dataset.

It should also be noted that our experiments did not address the dimensionality measures ($T2$, $T3$ and $T4$) and class imbalance measures ($C1$ and $C2$) as dimensionality and imbalances were constant in these experiments.

4. Experimental Setup

To test the applicability of the proposed meta-features, we conducted several experiments that compare β-frequencies to conventional meta-features on both artificial and real-world datasets.

We chose three popular algorithms that do not require complex hyper-parameter tuning to form the algorithm space $\mathcal{A}$ in our experiment: the k-Nearest Neighbours (kNN) (Cover and Hart, 1967; Dudani, 1978), Naive Bayes (NB) (Maron, 1961; Yang and Webb, 2001), and Decision Tree (DT) (Quinlan, 1993).

4.1. Meta-Labels

For each dataset, 10×10-fold cross-validation is used to evaluate the model generated by each algorithm in $\mathcal{A}$. We conduct t-tests to compare the performance of each pair of models. The null hypothesis is that the prediction accuracies of the pair of models do not significantly differ (i.e., we hypothesise that the difference in accuracies is 0). This is tested at a 5% level of significance ($\alpha = 0.05$). Meta-labels are assigned to each dataset based on the results of the t-tests. Table 2 shows all possible meta-labels and their corresponding algorithm superiority, as well as their corresponding pair-wise t-test results. Labels 0, 4, 5 and 6 indicate a draw between models generated by two or more algorithms. We discarded other possible combinations of t-test results as they contain contradictions.

Table 2: Meta-labels, algorithm superiority, and the winning model in pair-wise t-tests.

Meta-Label	Superior Algorithms	m_{kNN} vs m_{NB}	m_{kNN} vs m_{DT}	m_{NB} vs m_{DT}
Label 0	kNN, NB and DT	Draw	Draw	Draw
Label 1	kNN	m_{kNN}	m_{kNN}	–
Label 2	NB	m_{NB}	–	m_{NB}
Label 3	DT	–	m_{DT}	m_{DT}
Label 4	kNN and NB	Draw	m_{kNN}	m_{NB}
Label 5	kNN and DT	m_{kNN}	Draw	m_{NB}
Label 6	NB and DT	m_{NB}	m_{DT}	Draw

4.2. Meta-Features

Two sets of meta-features are used in our experiments. Set 1 contains the conventional meta-features, which consists of the classical and decision tree-based meta-features listed in Table 3, as well as the complexity meta-features listed in Table 1. Set 2 contains the 10-bin β-frequencies.

Table 3: Classical and decision tree-based meta-features.

Classical			**Decision Tree**	
ClassEnt	MutInfoMax	SkewnessMean	treewidth	ShortBranch
AttrEntMin	EquiAttr	SkewnessMax	treeheight	meanBranch
AttrEntMean	NoiseRatio	KurtosisMin	NumNode	devBranch
AttrEntMax	StdDevMin	KurtosisMean	NumLeave	maxAtt
JointEnt	StdDevMean	KurtosisMax	maxLevel	minAtt
MutInfoMin	StdDevMax		meanLevel	meanAtt
MutInfoMean	SkewnessMin		devLevel	devAtt

4.3. Preparation of Datasets

4.3.1. The Artificial Datasets

Each base-level dataset D that was generated consists of instances represented over two features and two classes, $D = \{\langle \mathbf{x}_i = \{x_{i,1}, x_{i,2}\}, y_i \in \{0, 1\}\rangle\}$. Each dataset consists of 625 instances evenly distributed in a 2-dimensional Euclidean plane of size 25×25 – i.e., for the i-th instance, $x_{i,1} = \lceil i/25 \rceil$ and $x_{i,2} = i \bmod 25$, where $i \in \{1, 2, ..., 625\}$.

To assign a class to each instance, we divided the feature space randomly into 2^t subsections of equal size, where $t \in \{1, ..., 9\}$. The instance at the centroid of each subsection is assigned a class (0 or 1) randomly. These labelled instances are then used as training instances, to build a model using each algorithm in the algorithm space $\mathcal{A}$. Each model is then used to predict and assign classes to the remaining unlabelled instances.

The above process is repeated over 10 iterations. Since $|\mathcal{A}| = 3$ and $|t| = 9$, a total of $3 \times 9 \times 10 = 270$ artificial datasets were produced. To ensure that the corpus of datasets is balanced in terms of meta-label distribution, we over-sampled the datasets with minority meta-labels, and eventually obtained 672 datasets in total. The frequencies of the base-level datasets with each meta-label are shown in Table 4.

Table 4: Frequencies of base-level artificial datasets of each meta-label.

Meta-label, l	0	1	2	3	4	5	6
Frequency, $N_l^{all.artificial}$	28	173	157	166	10	104	34

We randomly select the base-level datasets generated to form artificial meta-datasets, S. Additionally, to provide sufficient support, we generate each S such that $N_1^S, N_2^S, N_3^S \geq 10$ (where N_l^S corresponds to the number of instances in S with meta-label l), and $|S| \geq 100$.

We then calculate the imbalance ratio ($C2$) of each meta-dataset. The $C2$ values for the meta-datasets generated fell in the range $[0, 0.7746]$. We selected 25 meta-datasets in each of the $C2$ intervals: [0.0, 0.1), [0.1, 0.2) ... [0.7, 0.8), producing a total number of 200 artificial meta-datasets.

4.3.2. Datasets from the UCI Repository

We pre-process the UCI datasets (Dua and Graff, 2017) by scaling all continuous attributes to a range from 0 to 1, performing one-hot encoding on all discrete attributes, and removing irrelevant attributes such as the index.

For datasets with missing values, one different dataset is generated with each method: (i) removing all attributes with missing values, (ii) replacing the missing values in continuous attributes with the mean and the missing values in discrete attributes with the mode, and (iii) replacing the missing values in continuous attributes with the mean and treating the missing values for discrete attributes as a new attribute value.

To have a more diverse corpus of real-world datasets that are easily comparable, we utilise an error correcting output code (ECOC) mechanism (Dietterich and Bakiri, 1995) to convert multi-class datasets into binary ones. With all the above mentioned methods, we

were able to produce 7761 datasets from 190 original UCI datasets. The frequencies of the UCI datasets for each meta-label are shown in Table 5. We consider these 7761 base-level dataset as the UCI meta-dataset. The imbalance ratio ($C2$) of the UCI meta-dataset is 0.5662.

Table 5: Frequecies of base-level UCI datasets of each meta-label.

Meta-label, l	0	1	2	3	4	5	6
Frequency, $N_l^{all.UCI}$	124	528	484	5457	47	279	842

4.4. Overall Evaluation

For the final evaluation, we used the $allNN$ algorithm as the meta-learner, which is implemented using the `sklearn.neighbors.kNeighborsClassifier` (Pedregosa et al., 2011) with hyper-parameters $weight =$ `distance` and $n\_neighbor = |S| - 1$.

We then performed 10×10-fold cross-validation on each meta-dataset. M_{conv} denotes the algorithm selection model produced by training the meta-learner on the meta-dataset containing conventional meta-features, and M_β denotes the algorithm selection model produced by training the meta-learner on the meta-dataset containing β-frequencies. We compute the accuracy in each iteration of cross-validation in the following two ways:

- Weighted accuracy:

$$\frac{1}{|L|}\sum_{l\in L}\frac{|\{D\in S|Y_D = l \wedge M(D) = l\}|}{|\{D\in S|Y_D = l\}|}$$

  where L is the set of meta-labels and $M(D)$ is the prediction made by the model M.

- Unweighted accuracy:

$$\frac{|\{D\in S|M(D) = Y_D\}|}{|S|}$$

In meta-learning, it is unlikely that a sample of base-level datasets, i.e., the meta-dataset S, is representative of the unknown distribution of the population of all base-level datasets. Hence, we assume this distribution to be uniform. Consequently, we report both of the above accuracies for each experiment.

5. Results and Analysis

5.1. The Artificial Meta-Datasets

Table 6 shows the average accuracies of M_β and M_{conv} for each meta-label, as well as the overall weighted and unweighted accuracies, over artificial meta-datasets with different levels of imbalance. Figure 4 shows the percentage of wins on the y-axis and the imbalance intervals on the x-axis, based on t-tests. The proportion of M_{conv} wins is depicted in orange,

Table 6: Accuracy of M_β and M_{conv} over artificial meta-datasets.

Range of $C2$		Accuracy by Meta-Label							Overall Accuracy	
	Model	Label 0	Label 1	Label 2	Label 3	Label 4	Label 5	Label 6	Weighted	Unweighted
[0.0, 0.1)										
	M_β	14.42%	69.14%	57.78%	40.23%	0.00%	41.86%	11.92%	34.76%	52.58%
	M_{conv}	3.07%	73.83%	74.98%	73.87%	0.00%	55.23%	10.88%	43.26%	66.31%
[0.1, 0.2)										
	M_β	14.30%	74.41%	61.91%	48.41%	0.00%	29.67%	2.75%	33.58%	59.57%
	M_{conv}	0.00%	81.41%	75.86%	77.50%	0.00%	26.02%	0.98%	38.20%	71.33%
[0.2, 0.3)										
	M_β	14.27%	64.08%	36.43%	55.76%	0.00%	26.75%	3.96%	29.82%	62.75%
	M_{conv}	0.74%	71.00%	52.39%	76.37%	0.00%	21.97%	1.74%	33.78%	72.43%
[0.3, 0.4)										
	M_β	14.97%	71.42%	33.70%	34.01%	0.00%	16.41%	4.55%	25.60%	63.50%
	M_{conv}	7.38%	72.85%	39.63%	62.13%	0.00%	6.95%	6.70%	28.56%	69.54%
[0.4, 0.5)										
	M_β	11.62%	65.26%	36.17%	36.48%	0.00%	17.94%	2.54%	25.00%	68.17%
	M_{conv}	9.44%	61.76%	33.87%	59.09%	0.46%	8.58%	4.77%	26.28%	71.76%
[0.5, 0.6)										
	M_β	16.74%	52.42%	26.35%	43.23%	0.00%	9.15%	3.95%	22.08%	70.73%
	M_{conv}	5.42%	44.61%	26.03%	58.61%	0.00%	3.15%	1.33%	20.43%	72.76%
[0.6, 0.7)										
	M_β	10.27%	53.67%	18.30%	54.42%	0.00%	5.16%	1.09%	22.25%	76.59%
	M_{conv}	0.63%	44.08%	14.03%	70.17%	0.00%	3.41%	0.56%	20.87%	77.61%
[0.7, 0.8)										
	M_β	3.60%	56.34%	34.92%	30.00%	0.00%	0.80%	3.03%	21.15%	81.82%
	M_{conv}	0.00%	48.03%	35.77%	41.35%	0.00%	0.00%	0.40%	21.13%	82.69%

the proportion of M_β wins is depicted in blue, and the proportion of draws is depicted in grey.

Meta-labels 0, 5 and 6 are usually underrepresented because there are very few base-level datasets with these meta-labels available. We observe that M_β generally makes better predictions than M_{conv} for such base-level datasets. At high levels of imbalance ($C2 \geq 0.5$), M_β also makes better predictions for base-level datasets with meta-labels 1 and 2. The performance of both models on base-level datasets with meta-label 4 is extremely low, because there are very few (≤ 10) training instances available with that meta-label.

As imbalance increases, the performance of M_β improves and becomes more comparable to M_{conv}. Specifically, under high levels of imbalance ($C2 \geq 0.5$), the accuracy of M_β becomes higher than that of M_{conv}. The numbers of meta-datasets over which M_β wins is approximately equal to the number of meta-datasets over which M_{conv} wins.

The trend suggests that β-frequencies work better on more imbalanced meta-datasets – i.e., as meta-datasets becomes more imbalanced, it is more likely that β-frequencies would be of more value than conventional meta-features. Given that most real-world meta-datasets are imbalanced, β-frequencies may work as a viable substitute to the conventional meta-features.

The difference between the runtime for the generation of conventional meta-features and β-frequencies is significant. Over the artificial datasets, the conventional meta-features

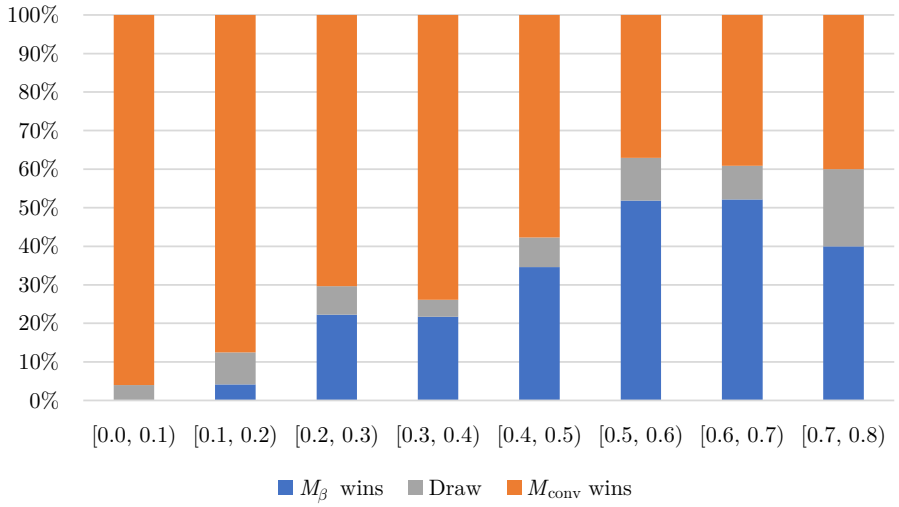

Figure 4: Composition of artificial meta-datasets in terms of t-test results.

required approximately 35 minutes to process, whereas the β-frequencies were computed in 21 seconds.

5.2. The UCI Datasets

The UCI datasets form a highly imbalanced meta-dataset (Table 5). Table 7 shows both the weighted and unweighted accuracy of M_β and M_{conv} over the UCI meta-dataset in each iteration of cross-validation (CV), as well as their overall accuracy score. The t-test between the weighted accuracy scores of M_β and M_{conv} yields a t-statistic of 127.7 and a p-value of 4.48×10^{-28}, indicating that M_β is a significantly superior algorithm selection model.

Table 7: Accuracy of M_β and M_{conv} over the UCI datasets.

Model	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10	Average
Weighted											
M_β	29.07%	28.72%	28.61%	28.62%	28.49%	28.64%	28.37%	28.97%	28.70%	28.79%	28.70%
M_{conv}	17.01%	17.00%	17.05%	17.08%	16.96%	16.91%	17.08%	16.51%	17.24%	17.19%	17.00%
Unweighted											
M_β	69.97%	69.73%	69.85%	69.97%	70.03%	69.95%	69.98%	70.00%	70.18%	69.91%	69.96%
M_{conv}	69.31%	69.32%	69.26%	69.35%	69.23%	69.36%	69.28%	69.22%	69.42%	69.30%	69.30%

The results suggest that β-frequencies, as meta-features, offer a viable and more efficient alternative to the conventional meta-features. In our experiments, M_{conv} uses over 50 pieces of information provided by all the conventional meta-features, while M_β achieves the same degree of accuracy using only 10 values based on the β-frequencies. The results further suggest that algorithm selection models based on β-frequencies may out-perform models based on conventional meta-features in terms of accuracy.

As expected, there was a significant difference between the runtime for the generation of conventional meta-features and β-frequencies. Over the UCI datasets, the conventional meta-features required approximately 2 weeks to process, whereas the β-frequencies were computed in 77 minutes.

6. Conclusion

In this paper, we provided a new perspective of landmarkers that consists of an algorithm and a metric that is measurable on the model generated by the algorithm. We then introduced a new metric, the distance-weighted class-homogeneous neighbourhood ratio, β. We have shown that the meta-features derived from β-values capture similar information to existing complexity measures, but at a lower computational cost. We then performed experiments over artificial and real-world datasets to test the applicability of β-frequencies. Our experimental results suggest two points: (i) when real-world repositories are utilised to build algorithm selection models, it is likely that an imbalanced meta-dataset is produced, in which case, the proposed meta-features have been shown to be significantly better; and (ii) should a balanced meta-dataset be produced, the results still suggest that the proposed meta-features are comparable to conventional meta-features, but at significantly lower computational cost. Considering the significantly lower cost of generation, β-frequencies may serve as viable meta-features for algorithm selection.

7. Future Work

We have provided evidence to show that β-frequencies are comparable to conventional meta-features. However, in order to further prove their viability, more base-level algorithms, meta-datasets, and meta-learners would be evaluated. Further, besides the 10-bin β-frequencies we proposed, more ways to utilise β-values as meta-features may be developed. Finally, in our current definition, β uses the target function $f^\star$ produced by $a^\star$. However, we may also extend this landmarker, $(a^\star, \beta)$, to (a, β), where a is an algorithm in the known algorithm space. In future work, we would like to examine the difference between $(a^\star, \beta)$ and (a, β) to provide insights for algorithm selection.

References

D. W. Aha. Generalizing from case studies: A case study. In *Machine Learning Proceedings 1992*, pages 1–10. Elsevier, 1992.

S. Ali and K. A. Smith. On learning algorithm selection for classification. *Applied Soft Computing*, 6(2):119–138, 2006.

P. Brazdil and C. Giraud-Carrier. Metalearning and algorithm selection: progress, state of the art and introduction to the 2018 special issue, 2018.

P. Brazdil and C. Soares. A comparison of ranking methods for classification algorithm selection. In *European Conference on Machine Learning*, pages 63–75. Springer, 2000.

P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to Data Mining*. Springer Science & Business Media, 2008.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

S. A. Dudani. The distance-weighted k-nearest neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, 8(4):311–313, 1978.

T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.

A. Kalousis and M. Hilario. Model selection via meta-learning: a comparative study. *International Journal on Artificial Intelligence Tools*, 10(04):525–554, 2001.

S. H. Kim, J. H. Jhong, J. J. Lee, and J. Y. Koo. Meta-analytic support vector machine for integrating multiple omics data. *BioData Mining*, 10(1):2, 2017.

B. Komer, J. Bergstra, and C. Eliasmith. Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In *ICML Workshop on AutoML*, pages 2825–2830, 2014.

J. W. Lee and C. Giraud-Carrier. Automatic selection of classification learning algorithms for data mining practitioners. *Intelligent Data Analysis*, 17(4):665–678, 2013.

A. C. Lorena, L. P. F. Garcia, J. Lehmann, M. C. P. Souto, and T. K. Ho. How complex is your classification problem? a survey on measuring classification complexity. *ACM Computing Surveys (CSUR)*, 52(5):1–34, 2019.

M. E. Maron. Automatic indexing: an experimental inquiry. *Journal of the ACM (JACM)*, 8(3):404–417, 1961.

D. Michie, D. J. Spiegelhalter, C. C. Taylor, et al. Machine learning. *Neural and Statistical Classification*, 13(1994):1–298, 1994.

N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

Y. Peng, P. A. Flach, C. Soares, and P. Brazdil. Improved dataset characterisation for meta-learning. In *International Conference on Discovery Science*, pages 141–152. Springer, 2002.

B. Pfahringer, H. Bensusan, and C. Giraud-Carrier. Meta-learning by landmarking various learning algorithms. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 743–750, 2000.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

J. R. Rice. The algorithm selection problem. In *Advances in Computers*, volume 15, pages 65–118. Elsevier, 1976.

P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

K. A. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, 41(1):6, 2009.

C. Soares and P Brazdil. Ranking classification algorithms with dataset selection: Using accuracy and time results. In *Proceedings of the Fifth International Workshop on Multistrategy Learning*, pages 249–260, 2000.

C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-weka: Automated selection and hyper-parameter optimization of classification algorithms. *CoRR, abs/1208.3719*, 2012.

L. Todorovski, H. Blockeel, and S. Dzeroski. Ranking with predictive clustering trees. In *European Conference on Machine Learning*, pages 444–455. Springer, 2002.

J. N. van Rijn, S. M. Abdulrahman, P. Brazdil, and J. Vanschoren. Fast algorithm selection using learning curves. In *International Symposium on Intelligent Data Analysis*, pages 298–309. Springer, 2015.

J. Vanschoren. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

L. Vendramin, R. J. G. B. Campello, and E. R. Hruschka. Relative clustering validity criteria: A comparative overview. *Statistical Analysis and Data Mining: the ASA Data Science Journal*, 3(4):209–235, 2010.

R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002.

D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8:1341–1390, 1996a.

D. H. Wolpert. The existence of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1391–1420, 1996b.

Y. Yang and G. I. Webb. Proportional k-interval discretization for naive-bayes classifiers. In *European Conference on Machine Learning*, pages 564–575. Springer, 2001.