Algorithm Selection for Classification Problems via Cluster-based Meta-features

Daren Ler*, Hongyu Teng†, Yu He† and Rahul Gidijala†
*Ministry of Education, Singapore
Email: daren_ler_shan_wen@moe.edu.sg
†National Junior College, Singapore
Email: {joey.teng.res, yuhe0208, rahul.gidijala}@gmail.com

***Abstract*—Meta-features describe the characteristics of the datasets to facilitate algorithm selection. This paper proposes a new set of meta-features based on clustering the instances within datasets. We propose the use of a greedy clustering algorithm, and evaluate the meta-features generated based on the learning curves produced by the Random Forest algorithm. We also compared the utility of the proposed meta-features against pre-existing meta-features described in the literature, and evaluated the applicability of these meta-features over a sample of UCI datasets. Our results show that these meta-features do indeed improve the performance when applied to the algorithm selection task.**

***Index Terms*—meta-learning; algorithm selection; cluster-based meta-features**

I. Introduction

The success of deep convolutional neural networks [1] have led to a series of breakthroughs for image classification [2], inspiring the proliferation of machine learning mechanisms. There is thus an increasing demand for easy-to-use, off-the-shelf methods that do not require machine learning expertise [3]. In order to function as a blackbox, i.e. to solve classification problems without human intervention, additional research on Automatic Machine Learning (AutoML) is required. Examples of AutoML include the Java-based WEKA tool [4], and the Python-based scikit-learn module [5].

According to the 'No Free Lunch' (NFL) theorems [6], no single learning algorithm can uniformly outperform other algorithms over all learning tasks [7]. Consequently, in order to facilitate effective AutoML, an algorithm selection mechanism is typically necessary.

In [8], a framework was proposed that utilises measurable features (i.e., meta-features) to describe datasets, which are in turn used to facilitate algorithm selection (meta-learning). Since that work, the field of meta-learning has evolved to utilise various forms of meta-features and learning algorithms (as meta-learners - i.e., the mechanisms that perform algorithm selection) to solve the problem of algorithm selection [9]–[13]. Furthermore, meta-learning has expanded to solve many different types of problems, as described in various review papers [11], [14]–[16].

This paper explores a method of measuring meta-features that is based on clustering, which is largely motivated by the T1 Complexity measure proposed by [17]. The idea is to investigate the possibility of capturing data characteristics by measuring the properties of homogeneous clusters - e.g. number, size and volume of clusters, and cluster proportion by class. In our work, we generate these meta-features over various classification tasks and then apply them to algorithm selection over decision trees and random forests. This work is similar to the use of meta-features based on of decision tree properties that was proposed by [18].

In the next section of this paper, a more detailed description of related work is provided. This is then followed by a description and evaluation of the proposed cluster-based meta-features over artificial datasets. The paper then continues with a description of the specific algorithm that is adopted to generate homogeneous clusters, and proceeds to describe further experiments, their results, and analysis, before concluding with a summary of our findings and future work.

II. Related Work: Meta-Learning for Algorithm Selection

One aspect of meta-learning is concerned with the selection of an appropriate algorithm given a learning problem. It aims to find functions that map datasets to algorithm performance (e.g. predictive accuracies, execution time, etc). Such mappings require the determination of dataset characteristics, or meta-features, which should be indicative of algorithm performance [19]. Although various meta-features have been proposed, there is still much room for research.

The algorithm selection framework defined by Rice is adapted to derive the framework depicted in Fig. 1. In this framework, we begin with a set of datasets $\mathbb{D}$, in this case with various datasets from the UCI dataset repository [20]. For each dataset $D_i \in \mathbb{D}$, the performance metric P is evaluated for each learning algorithm $L_i \in \mathbb{L}$. In this paper, we use the predictive accuracy of the model as P, and the algorithms used are Random Forest and Decision Tree classifiers. The characteristics of each dataset are captured by generating meta-features F_i. A meta-dataset Ω is constructed using these $F_i \in \mathbb{F}$, and the meta-labels $\nu_i \in \nu$ assigned based on P, with each record Ω_i defined as a conjunction of F_i and ν_i. A meta-learning algorithm L^* is then trained over the meta-dataset Ω, to produce a model M. The model M will be used to predict an algorithm for a new dataset, based on the meta-features generated from it. The meta-learning algorithm used in this paper is the k-Nearest Neighbours algorithm.

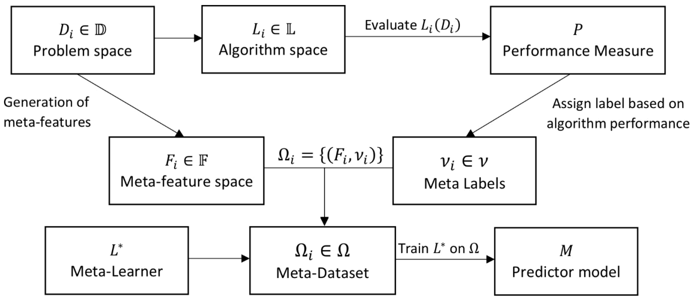
$D_i \in \mathbb{D}$ Problem space
$L_i \in \mathbb{L}$ Algorithm space
Evaluate $L_i(D_i)$
P Performance Measure
Generation of meta-features
Assign label based on algorithm performance
$F_i \in \mathbb{F}$ Meta-feature space
$\Omega_i = \{(F_i, v_i)\}$
$v_i \in v$ Meta Labels
L^* Meta-Learner
$\Omega_i \in \Omega$ Meta-Dataset
Train L^* on Ω
M Predictor model

Fig. 1. Algorithm selection framework

A. Dataset Characterisation: Meta-features

The choice of meta-features typically depends on the problem domain as well as the chosen algorithms. Meta-features must be chosen such that any known structural properties and complexities of the problem are exposed, and any known advantages or limitations of the different algorithms are captured [14].

The first meta-features utilised include the number of instances, number of classes, number of prototypes per class, number of relevant and irrelevant attributes, and the distribution range of the instances and prototypes [21]. In [9], these features were extended to include statistical measures (e.g. mean and variance of instance attributes), and information theory measures (e.g. entropy).

A number of studies then extended the set of meta-features to include the structural characteristics of decision tree models for each dataset (such as the number of nodes and leaves, width and depth of the tree, distribution of attributes and nodes in the tree, etc.) and investigating the optimal set of features to include in the model [22], [23]. In a similar vein, the idea of landmarking seeks to generate computationally inexpensive features based on the performance of models generated by simple learning algorithms [24].

Other proposed meta-features include computational complexity and time [25], and the index of dispersion, center of gravity, and maximum and minimum eigenvalues of the covariance matrix [26]. More recently, [27] also proposed more diverse features to describe datasets for the purpose of optimising the choice of algorithm hyperparameters, while [28] investigated complexity-based meta-features for regression problems.

More holistic research was also recently conducted using a set of 509 features derived from the eight feature sets: simple, statistical, information theoretic, landmarking, model-based, concept characterisation, complexity, and itemset and association rules-based [29].

Alternatively, the work in [28] used measures devoted to estimating the complexity of the function that should be fitted to the data in regression problems, with three meta-learning setups: (i) predicting the regression function type of some synthetic datasets; (ii) tuning of parameter values of support vector regressors; and (iii) prediction of the performance of various regressors for a given dataset.

III. Cluster-based Meta-features

A. The Proposed Meta-Features

Intuitively speaking, one may assume that dataset complexity is strongly correlated to the accuracy of the classifiers trained using that dataset. This notion is central to the thesis proposed within this paper: that information concerning the complexity of a dataset would serve as good meta-features.

One problem with this thesis is that the measurement of complexity is typically associated with one or a family of models/functions (e.g., VC dimension). However, we wish to consider less model-centric measures for complexity, and instead focus on a general notion of dataset complexity (i.e., one that is model-independent). Our proposed solution is to approximate the minimum number of spherical homogeneous clusters that are required to cluster all the instances within a given dataset, and then to calculate meta-features from the clusters formed.

Definition 1: Non-overlapping Spherical Cluster. Given a dataset, $\mathbb{D}_i$, let $\mathbb{C}_j$ correspond to a set of instances $\{(X_1, y_1\}, \ldots, (X_k, y_k)\} \subseteq \mathbb{D}_i$, let $\overline{X_j}$ correspond to the coordinates of the centroid of $\mathbb{C}_j$, and let (X'_j, y'_j) correspond to the instance in $\mathbb{C}_j$ that is most distant to $\overline{X_j}$. $\mathbb{C}_j$ is a non-overlapping spherical cluster iff $\{(X_a, y_a)|(X_a, y_a) \in \mathbb{D}_i \backslash \mathbb{C}_j \wedge \|X_a, \overline{X_j}\| \leq \|X'_j, \overline{X_j}\|\} = \emptyset$. More loosely, a non-overlapping spherical cluster is defined as a set of instances from a given dataset such that the sphere surrounding those instances does not also encompass any other instances from the dataset; i.e., the cluster only overlaps the instances it contains and no others.

Definition 2: Homogeneous Non-overlapping Spherical Cluster. A non-overlapping spherical cluster, $\mathbb{C}_i$, is a homogeneous non-overlapping spherical cluster iff for each instance $(X_j, y_j) \in \mathbb{C}_i, y_j = \overline{y}$ (where $\overline{y}$ corresponds to the class label associated with the cluster $\mathbb{C}_i$). More loosely, a spherical non-overlapping homogeneous cluster corresponds to a non-overlapping spherical cluster which only contains instances with the same class label.

Definition 3: Dataset Coverage. A set of spherical non-overlapping homogeneous clusters, $\{\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_m\}$, covers a dataset $\mathbb{D}_i$, iff $\bigcup_{j=1}^{m} \mathbb{C}_j = \mathbb{D}_i$. More loosely, a set of spherical non-overlapping homogeneous clusters covers a given dataset $\mathbb{D}_i$ only if each instance in $\mathbb{D}_i$ belongs to exactly one cluster in that set.

Extending from the basic thesis described above, we propose that meta-features generated from the set of homogeneous non-overlapping spherical clusters that: (i) covers a dataset, and (ii) has the minimum possible cardinality (i.e., minimum possible number of clusters), will be highly indicative of the complexity of the dataset since it approximates the minimum number of linear separators that would be required to model that dataset. However, since it might be too expensive to calculate the exact set of clusters with the lowest cardinality, we will instead use an approximation of this minimum.

Thus far, complexity has only been loosely defined. What then is exactly meant by the complexity of a dataset? The following three types of complexities are addressed by the proposed cluster-based meta-features.

1) the complexity corresponding to the imbalance of membership of instances
2) the complexity corresponding to the concentration of the instances in each class
3) the complexity corresponding to the wide spread of the decision boundary

Accordingly, the specific cluster-based meta-features proposed are as follows.

1) **PurityRatio**(y_i): the ratio of the number of clusters in NumClusters with class y_i to the total number of clusters calculated. The larger the PurityRatio is of a dataset, the more complex the dataset is. Range: $[0, 1]$.
2) **SizeDist**: the distribution of clusters based on the number of instances in them; approximated by a frequency histogram normalised by the total number of clusters over 10 equal intervals (where each interval reports the number of clusters containing the numbers of instances defined by the interval values). The more the histogram skewed to the right, the more complex the dataset is. Range $[0, 1]$ for each normalised frequency in its respective interval.
3) **VolumeDist**: the distribution of the clusters based on cluster volume; again, approximated by a frequency histogram normalised by the total number of clusters over 10 equal intervals (where each interval reports the number of clusters with volumes defined by the interval values). The more the histogram skewed to the right, the more complex the dataset is. Range $[0, 1]$ for each normalised frequency in its respective interval.

It should be noted that while the proposed clusters are similar to the T1 Complexity Measure proposed by [17], they are not the same. With the T1 complexity measure proposed by [17], the metric (i.e., meta-feature) corresponds to a single value that integrates 2 forms of complexity (i.e., decision boundary spread and instance concentration). However, there is a clear separation between these 2 forms of complexity within the meta-features proposed in this paper. The distribution of cluster volumes reflects complexity brought about by decision boundary spread, whereas the distribution of cluster sizes directly relates to the instance concentration.

B. The Clustering Algorithm

As noted in the previous section, the proposed meta-features are to be computed based on a set of homogeneous non-overlapping spherical clusters with minimum (or approximately minimum) cardinality. However, as we will show in the following subsection, the generation of such a set of clusters can the prohibitively expensive.

1) Brute Force Variant: Initially, we considered utilising a brute force variant of a clustering algorithm that would ensure that we had found the smallest set of clusters. Essentially, we may pose such a clustering problem as a partition problem: i.e., to distribute $n = |\mathbb{D}_i|$ distinct objects (i.e., the instance in a given dataset) into 1 to n identical, non-empty bins (i.e., the potential clusters). Consequently, for a given dataset, $\mathbb{D}_i$, where $|\mathbb{D}_i| = n$, the total number of possible partitions is given by the n^{th} Bell number:

$$B_n = \sum_{k=0}^{n} S(n, k), \tag{1}$$

where $S(n, k)$ is a Stirling number of the second kind [30].

For each such partition, we would then have to perform two tests on each bin: (1) determine if the bin may be defined as a non-overlapping spherical cluster (i.e., ensuring that the sphere encasing the instances within the bin does not also include any instances that are not part of that bin), and if so, then, (2) determine if the bin may further be defined as a homogeneous non-overlapping spherical cluster (i.e., if all the instances in that bin have same class label). The smallest set of bins/clusters that satisfy the above two tests would thus correspond to the smallest set of homogeneous non-overlapping spherical clusters.

```
for $k \leftarrow 1$ to $dataset.size$ do
    $partitions \leftarrow$ GET_ALL_PARTITIONS($dataset$, $k$)
    // returns all possible sets of $k$
       non-empty clusters
    for $i \leftarrow 1$ to $partitions.size$ do
        if IS_NON_OVERLAPPING($partitions$, $partitions[i]$)
         and IS_HOMOGENEOUS($partitions[i]$) then
            return $partitions[i]$ // the optimal
               (i.e., smallest) set of clusters
        end
    end
end
```

Algorithm 1: BRUTE_FORCE_CLUSTER($dataset$)

While this algorithm ensures an optimal solution, is has $O(n^n)$ runtime complexity [31], and is thus impractical, especially with larger datasets.

2) Greedy Variant: To significantly reduce the complexity of generating the smallest set of homogeneous non-overlapping spherical clusters, we instead adopt a greedy approach, which approximates the optimal solution.

This greedy variant starts with each instance as a singleton cluster (which, of course, will correspond to a valid homogeneous non-overlapping spherical cluster). It then iterates through the clusters, merging the current cluster with its closest adjacent cluster (based on cluster centroid distances) with the same class label. An additional check is also required to ensure that any newly combined cluster does not overlap with any other existing clusters. This is done by checking that the distance between two cluster centroids is greater than the sum of the radii of those two clusters. If the newly proposed combined cluster overlaps any of the existing clusters, then the combination is not performed. The algorithm continually loops over all existing clusters until no new clusters are combined over an entire iteration of the current set of clusters.

```
$partition \leftarrow$ INITIALISE_SINGLETON_CLUSTER($dataset$)
repeat
    $oldsize \leftarrow partitions.size$
    $i \leftarrow 1$
    while $i \leq partition.size$ do
        $closest \leftarrow$ FIND_CLOSEST_CLUSTER($partition$,
          $partition[i]$)
        // excludes target in search

        $merged \leftarrow$ MERGE_CLUSTERS($partition[i]$, $closest$)
        if IS_NON_OVERLAPPING($merged$, $partition$)
         and IS_HOMOGENEOUS($merged$) then
            $partition[i] \leftarrow merged$
            $partition.remove(closest)$
            // no $i$ adjust since $partition[i]$ was
               removed
            // and index of $closest$ must be
               greater than $i$
        end
        else
            $i \leftarrow i + 1$
        end
    end
until $oldsize \leq partition.size$ // $size - 1$ per merge

return $partition$
```

Algorithm 2: GREEDY_CLUSTER($dataset$)

The time complexity of this greedy variant is $O(n^3)$, where $n = |\mathbb{D}_i|$. As compared to the brute force variant, it is a far more practical solution, and as such, is the algorithm we adopt in generating an approximately minimal set of homogeneous non-overlapping spherical clusters.

IV. Experimental Design

To approximate the complexity of a dataset, we use the characteristics of the learning curve derived from the application of a learning algorithm. More specifically, the Random Forest algorithm. We use this algorithm as it has been shown to be appropriate in most cases [32].

Intuitively speaking, if the slope of a learning curve is more gradual, we assume that algorithm is capable of exploiting more instances to continually improve its accuracy (conversely, with a steep learning curve, the algorithm will no longer be able to improve accuracy beyond a certain training set size) – i.e., the shape of the learning curve suggests the complexity of the dataset. It is believed that this measure is generally applicable regardless of the choice of learning algorithm, i.e. a more complicated classification problem for one algorithm would similarly be more complicated to another learning algorithm.

So as not to incur high experimental complexity, only limited points on the learning curve are calculated and plotted. A continuous analytical expression for a curve would be easier for further investigation than discrete data points. Previous work has suggested a good model fit to learning curve data [33]:

$$\widehat{acc}(n) = a - bn^{-\alpha} \tag{2}$$

where $\widehat{acc}(n)$ indicates the estimated accuracy of the model using a sample with size n.

To compare the relative complexity of datasets via learning curves, we propose to use two quantities: α from equation 2 above, and the inverse of area under the graph, A^{-1}, where:

$$A = \int_0^{100} \hat{acc}(n)\mathrm{d}n \tag{3}$$

More specifically, we hypothesise that a larger value of α, and/or a larger value of A^{-1} would be obtained for a more complex dataset.

We verify this hypothesis empirically by experimenting with the following artificial datasets. Four sets of datasets, each comprising 200 binary datasets, are generated. Each dataset contains 30×30 evenly distributed instances in a 2-D instance space. The datasets in each set have different configurations, which are defined as follows.

- Set 1 (kNN-based): The instances are categorised based on the sum of weighted distances to the initially randomly labelled (pivot) instances, which is very similar to the all-neighbours kNN method. This is depicted in Fig. 2(a).

  Let m be the number of pivot instances. 20 groups of datasets are generated with $m \in \{5, 10, 15, ..., 100\}$. For each value of m, a group of 10 datasets are generated with m randomly generated pivot points in the space, $\lfloor \frac{m}{2} \rfloor$ pivot points belong to the negative class, and $\lceil \frac{m}{2} \rceil$ pivot points belong to the positive class. The class memberships of the 900 instances are determined using all-neighbours kNN method taking these m pivot points as labelled training samples. Note that the pivot points will not be included in the final dataset, so the total number of instances for each dataset is kept at 900.

- Set 2 and 3: Several linear separators that all pass through the centre or bottom-left corner, are used to label the instances. These are depicted in Fig. 2(b), and Fig. 2(c) respectively.

  Let m be the number of linear separators in Set 2 and 3. 20 groups of datasets are generated with $m \in \{5, 10, 15, ..., 100\}$. For each value of m, a group of 10 datasets are generated with m randomly generated linear separators in the space. Then, classes are assigned to the 900 instances. The regions of instances are assigned with positive, then negative classes in an alternating pattern in an anti-clockwise direction with the region at the top-left corner always belonging to the positive class.

- Set 4: The instances are separated by linear separators that are orthogonal to the axises. This is depicted in Fig. 2(d).

  Let m be the number of linear separators. 20 groups of datasets are generated with $m \in \{5, 10, 15, ..., 100\}$. For each value of m, a group of 10 datasets are generated with m randomly generated linear separators in the space, where $\lfloor \frac{m}{2} \rfloor$ of them are in the horizontal direction and $\lceil \frac{m}{2} \rceil$ of them are in the vertical direction. Then, classes are assigned to the 900 instances. The regions of instances are assigned with positive, then negative classes in an alternating pattern such that no two adjacent regions (i.e., regions with a common side) have the same class membership. The region at top-left corner always belongs to the positive class.

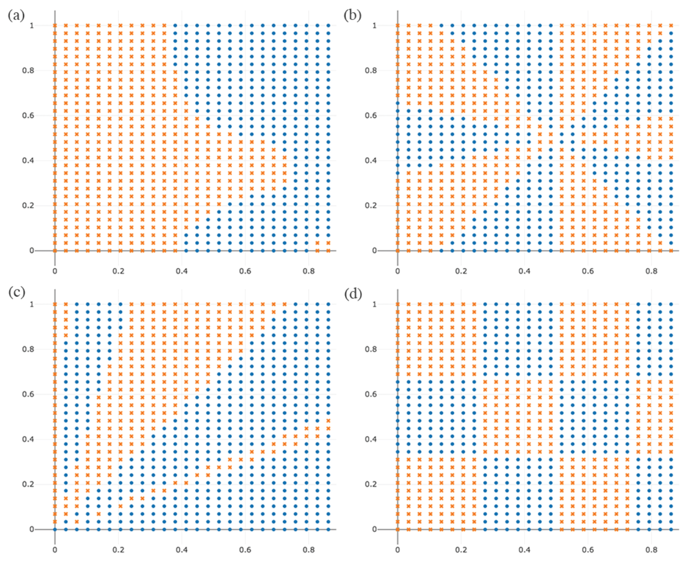

Fig. 2. Artificial datasets generated by (a) all-neighbours kNN method; (b) linear separators with common intersection at the centre; (c) linear separators with common intersection at the corner; (d) linear separators that are orthogonal to the axises. Sets 1 and 3 have randomised parameters whereas Sets 2 and 4 have parameters to ensure the even distribution.

Experiments on all 800 artificial 2D binary datasets are conducted to investigate the correlation between the complexity of the dataset measured by two quantities above and the meta-features we computed based on clustering. Another set of experiments on over 184 UCI datasets [20] are conducted to verify the linear correlation between each of our proposed meta-features and the quantities obtained from the learning curve.

All learning algorithm used within this and latter experiments were taken from the Python scikit-learn library [34].

V. Results and Discussion for Measuring Complexity

Based on the results of our experimentation, as shown in Table I, we can infer that a dataset generated with a larger m will exhibit higher complexity. From Table I, the high R^2 values associated with the inverse of area under the learning curve (A^{-1}) suggests that it is a good indicator of dataset complexity.

TABLE I
The R^2 values of the learning curve characteristics, each versus the number of separators/labelled instances.

Generation Method	R^2 of α to m	R^2 of A^-1 to m
kNN-based	0.36258760751231417	0.6951633184415388
Intercept at centre	0.0077197377696427575	0.455378285976787
Intercept at corner	0.00078373951977795139	0.6663951649278638
Orthogonal separators	0.0036320190100626692	0.27621223117160626

It should be noted that although a typical learning curve has a non-decreasing monotonous shape (as shown in Fig. 3a), learning curves with quadratic-like shape (as shown in Fig. 3b) do exist.

All sets of experiments have several cases where a learning curve has a decreasing trend (Fig. 3c). Additionally, if the expression poorly fits the curve (based on the criteria: $R^2 < 0.5$ between the predicted value and actual value), it will be categorised as indeterminate (Fig. 3d).

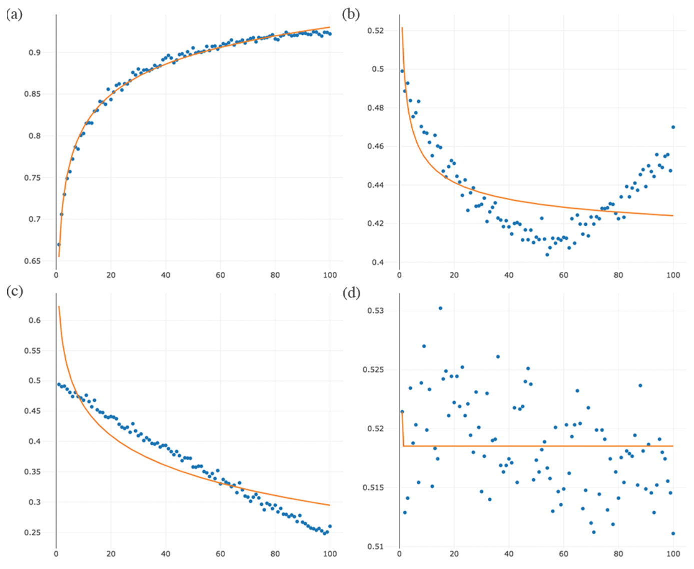

Fig. 3. Different types of learning curve shapes: (a) Typical shape (dataset is generated by linear separators with common interception at the centre); (b) Quadratic-like shape (dataset is generated by orthogonal linear separators); (c) Decreasing trend (dataset is generated by orthogonal linear separators); (d) Indeterminate shape.

Table II shows the frequency of each learning curve category over the different sets. It should be further noted that the indeterminate shaped learning curves are generally observed when the accuracy does not change much (± 0.2), which suggests a target model that random forest algorithm is unable to further specify (i.e., it is unable to further improve accuracy with additional instances).

Table II further allows us to conclude that the majority of the datasets will have a typical learning curve. Consequently, we will consider only these datasets in further analysis.

TABLE II
FREQUENCIES OF THE TYPES OF LEARNING CURVE SHAPES OBSERVED IN DIFFERENT EXPERIMENTAL SETS.

Datasets	Monotonous Increasing[a]	Quadratic-like	Decreasing	Indeterminate
kNN-based	200 (100%)	0 (0%)	0 (0%)	0 (0%)
Intercept at centre	169 (84.5%)	6 (3%)	17 (8.5%)	8 (4%)
Intercept at corner	183 (91.5%)	0 (0%)	10 (5%)	7 (3.5%)
Orthogonal separators	145 (72.5%)	19 (9.5%)	27 (13.5%)	9 (4.5%)
UCI datasets	167 (90.8%)	0 (0%)	6 (3.3%)	11 (5.9%)
Total	697 (87.76%)	25 (2.55%)	54 (6.12%)	24 (3.57%)

[a]In this paper, monotonous Increasing refers to the normal (i.e., the typically expected) shape for a learning curve. (refer to Figure 3a)

Based on the data shown in the Table III, there is a strong correlation between the number of clusters obtained and inverse of the area under learning curve. This implies that the number of clusters may be a good indicator for the complexity of the dataset, and hence will be useful in algorithm selection.

VI. EXPERIMENTAL EVALUATION

To study the usefulness of the proposed meta-features, we compare them to pre-existing meta-features described in the literature. The experiments are conducted on the meta-features in sets, based on their categorical names (as defined in the literature). The different sets of meta-features are as follows:

- **Set A:** Classical meta-features (statistical and information-theoretic measures)
- **Set B:** Decision-tree based meta-features
- **Set C:** The proposed cluster based meta-features

And combinations of these 3 main sets, which are as follows:

- **Set A+B:** Classical and decision-tree based
- **Set A+C:** Classical and cluster based
- **Set B+C:** Decision-tree based and cluster based
- **Set A+B+C:** All sets combined

We omit general meta-features like the number of instances, number of features etc., due to their inability to adequately capture dataset complexity. We also do not use all of the possible subsets of meta-features as there are too many (2^{55} possible subsets) and it would be too computationally expensive to analyse all of them. Hence, in this analysis, we only utilise traditional subsets as defined in the literature. The full list of each set is in Table IV.

Along with the meta-features generated over the 184 UCI datasets, the labels corresponding to the algorithm superiority over these datasets were computed based on a two-tailed paired t-test over the 10×10-fold cross-validation, over both the random forest and decision tree algorithms.

TABLE III
R^2 VALUES BETWEEN CHARACTERISTICS OF LEARNING CURVES AND CLUSTER BASED META-FEATURES OBTAINED FROM THE 800 ARTIFICIAL 2D BINARY DATASETS.

Quantity	Coefficient	Area Inverse
NumberClusters[a]	0.0004512853465366110	0.8398898181780040
PurityRatio(Positive)[b]	0.000000150624085258312	0.0010316953362306
SizeDist[c].01	0.006919088170462090	0.7700741060274190
SizeDist[c].02	0.008700482534033450	0.005827685905972380
SizeDist[c].03	0.0014271446847680800	0.2719227779611430
SizeDist[c].04	0.0005372091959385090	0.12766135777474700
SizeDist[c].05	0.002446738078102080	0.09944580981798230
SizeDist[c].06	0.009304293303490010	0.06363632359615560
SizeDist[c].07	0.001132177621296050	0.24887597954949500
SizeDist[c].08	0.0000827812452891725	0.024305808790407000
SizeDist[c].09	0.00010490983986294500	0.07665109423823190
SizeDist[c].10	0.0000811134806924371	0.04902942310935510
VolumeDist[d].01	0.0006287189742214160	0.6906906097562030
VolumeDist[d].02	0.00044687020297598000	0.22836562993448000
VolumeDist[d].03	0.000000456513926658977	0.14915897898301200
VolumeDist[d].04	0.0010592752003745100	0.28244248913689700
VolumeDist[d].05	0.000850548310300262	0.07338897140121900
VolumeDist[d].06	0.0019436470768623000	0.029686052259901200
VolumeDist[d].07	0.0006857318103772070	0.07173712739034330
VolumeDist[d].08	0.01638935253703630	0.008646957452851750
VolumeDist[d].09	0.0002627744024574980	0.0007510706141766470
VolumeDist[d].10	0.000000228840989209466	0.0778524785091906

[a]The approximated minimum number of clusters required to cover a dataset

[b]Proportion of positive clusters over all clusters (not normalised)

[c]The non-normalised frequency of clusters based on their normalised radii: divided into 10 even intervals

[d]The non-normalised frequency of clusters based on their normalised sizes: divided into 10 even intervals

The utility of a meta-feature set is defined in terms of the predictive accuracy of a meta-classifier built on those meta-features to predict algorithm superiority. The meta-learning algorithm used in this paper is the k-Nearest Neighbours (kNN) algorithm.

Evaluation is done by using stratified 10x10 cross-validation. Each model was trained using a wrapper-based method to select the value of k i.e. we selected the value of k by iterating from 1 to 20; this resulted in models ranging from 1-NN, 2-NN, ..., 20-NN.

For each meta-feature set, the best performing kNN model was identified as having the greatest mean predictive accuracy on a validation set. We then compared the similarities between the predictions of each meta-feature set pair, based on mutual information.

The results of these experiments are shown and discussed

TABLE IV
LIST OF META-FEATURES UTILISED.

Set	Features	Set	Features
Set A (19)	ClassEnt AttrEntMin AttrEntMean AttrEntMax JointEnt MutInfoMin MutInfoMean MutInfoMax EquiAttr NoiseRatio StandardDevMin StandardDevMean StandardDevMax SkewnessMin SkewnessMean SkewnessMax KurtosisMin KurtosisMean KurtosisMax	Set B (14)	treewidth treeheight NumNode NumLeave maxLevel meanLevel devLevel ShortBranch meanBranch devBranch maxAtt minAtt meanAtt devAtt
		Set C (22)	NumberClusters PurityRatio(Positive) SizeDist (Intervals of 10) VolumeDist (Intervals of 10)

in the next section.

VII. RESULTS AND ANALYSIS

Table V lists the mean accuracies of the best-performing kNN model for each meta-feature set.

As expected, the meta-feature set including all meta-features produced the most accurate model for algorithm selection. What is the more interesting is that, in isolation, the cluster based meta-features do not produce a very accurate model. However, in conjunction with other meta-feature sets, the cluster-based meta-features improved accuracy, with the greatest improvement observed in Set A+C. The mean accuracy of Set A+C was higher than A+B, suggesting that cluster based meta-features were more useful in conjunction with other measures than the decision-tree based meta-features. The difference is small. However, to support this hypothesis, further experimentation is required. The set of all meta-features was the best set, with a significant improvement in accuracy over the second best model. It achieved an accuracy value of almost 86%.

Table VI shows the accuracies of each meta-feature set, with the datasets broken down into categories based on the learning rates of the random forest algorithm (refer to Section 5). Upon analysing the individual learning curves, the datasets in the 'indeterminate' category are very similar in nature to the 'normal' category, and potentially, may be combined. The difference could possibly be explained by the granularity used over training set sizes. If this granularity is set too high it may not capture the trend in detail. This explains the similarity in the accuracies of the two categories. However, the datasets in the 'decreasing' category exhibit a clear difficulty in predicting algorithm superiority. Consequently, this would account for a significant amount of the overall error with each meta-feature set. This table also shows that datasets falling under the 'normal' category, which is the most typical form, are the easiest to predict. Further investigation into the characteristics of these categories may provide insight into their effects on classifier performance and algorithm prediction.

TABLE V
BEST K AND CORRESPONDING ACCURACIES FOR EACH META-FEATURE SET

Meta-Dataset	Best K	Corresponding Accuracy
A	1	0.78725 ± 0.00842
B	12	0.76477 ± 0.00727
C	4	0.72428 ± 0.01251
AB	5	0.83371 ± 0.00795
AC	5	0.83920 ± 0.00833
BC	1	0.77293 ± 0.01360
ABC	5	0.85759 ± 0.00895

TABLE VI
THE BREAKDOWN OF ACCURACIES BASED ON LEARNING CURVE CATEGORIES.

Category	All datasets	Normal
A	0.78725 ± 0.00842	0.80060 ± 0.00507
B	0.76477 ± 0.00727	0.78982 ± 0.00792
C	0.72428 ± 0.01251	0.74012 ± 0.01304
AB	0.83371 ± 0.00795	0.86048 ± 0.00652
AC	0.83920 ± 0.00833	0.85868 ± 0.00971
BC	0.77293 ± 0.01360	0.78683 ± 0.01336
ABC	0.85759 ± 0.00895	0.87844 ± 0.00920
Category	Decreasing	Indeterminate
A	0.31667 ± 0.09723	0.83636 ± 0.03939
B	0.21667 ± 0.11561	0.69091 ± 0.04825
C	0.31667 ± 0.12638	0.70000 ± 0.04513
AB	0.25000 ± 0.12111	0.75455 ± 0.04513
AC	0.31667 ± 0.12638	0.81818 ± 0.00000
BC	0.33333 ± 0.11419	0.80909 ± 0.02954
ABC	0.36667 ± 0.17690	0.81818 ± 0.00000

Table VII shows the mutual information values between the predictions of each pair of meta-feature sets. The low mutual information between Set C and Sets A and B imply that the models trained using the cluster based meta-features are diverse in comparison to the other two, and vice versa. This, combined with the differences in accuracies, suggests that cluster based meta-features are responsible for the improvement in accuracy observed on the model trained using Set A+B+C. This may imply that cluster based meta-features may potentially be useful in augmenting existing meta-feature sets.

TABLE VII
MUTUAL INFORMATION BETWEEN PREDICTIONS OF EACH PAIR OF META-FEATURE SETS

Meta-feature Set Pair	Mutual Information	Meta-feature Set Pair	Mutual Information
A vs B	0.039486402	C vs AB	0.076676467
A vs C	0.051302072	C vs AC	0.178249324
A vs AB	0.076202889	C vs BC	0.201737995
A vs AC	0.087205307	C vs ABC	0.077700597
A vs BC	0.034029822	AB vs AC	0.151260003
A vs ABC	0.044589671	AB vs BC	0.117304218
B vs C	0.060159643	AB vs ABC	0.218018331
B vs AB	0.265991136	AC vs BC	0.110424136
B vs AC	0.066965286	AC vs ABC	0.236226749
B vs BC	0.136978104	BC vs ABC	0.193795588
B vs ABC	0.131795987		

We plot each dataset within an expertise space, with the accuracy of the decision tree algorithm against the accuracy of the random forest algorithm. The points are marked according to the number of meta-feature sets that we're able to produce a model that was able to accurately predict algorithm superiority. This is depicted in Fig. 4.

Fig. 4 shows that most of the datasets lie on the line dividing the space diagonally (i.e., the points where the accuracies over the two algorithms are similar). We may infer that this is the case because the datasets are not very complex, and, the accuracies of both models would not differ significantly on most of them. The points in Fig. 4 to take note of are those deviating significantly away from this line. These points indicate a clear superiority of one algorithm over the other. Again, as expected, the random forest algorithm, as an ensemble (i.e., more complex algorithm), outperforms the decision tree on most of the datasets. Consequently, there are few points lying in the region of decision tree superiority, and these points tend to be close to the line. This supports our assumption that the random forest algorithm is, in general, better.

The more important observation is that the datasets with the decision tree algorithm performing better, have significantly fewer models able to accurately predict them (0 or 1 models accurately predicting these datasets in most cases). In contrast, the datasets with random forest superiority had more models able to accurately predict them (6 or 7 in most cases). This suggests that further investigation into the relationship between the underlying characteristics of these (outlier datasets) datasets is warranted.

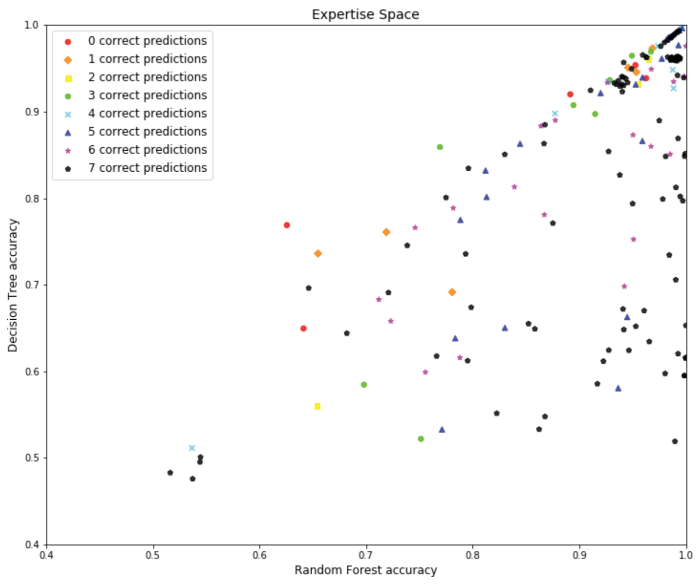

Fig. 4. Expertise space graph, with points marked on meta-feature sets predictions

VIII. Conclusion

The generation of appropriate and useful meta-features is an important part of the algorithm selection process. In this paper, we introduced a new set of meta-features and the algorithm to generate them. This method is based on the hypothesis that clusters induced on a dataset will help to capture the complexity of the dataset. This is done by extracting the features of the clusters induced on a dataset using a greedy approach. We propose a total of 22 cluster-based meta-features to capture the properties of the cluster model. We found a strong correlation between the characteristics of the learning curves and the complexity of the dataset. Based on our initial experimentation, we also found that the cluster-based meta-features are good indicators of the dataset complexity. From our analysis, we also found that the utility of the cluster-based meta-features in isolation is limited. However, they are useful in augmenting existing meta-feature sets.

IX. Future Work

Future work includes the following:

1) Formalise theories that justify the utility of the proposed meta-features. In particular, we should seek to formalise the relationship between the specific forms of complexity (balance, spread of decision boundaries, and concentration) and the specific meta-features.
2) Explore more datasets in order to derive a taxonomy based on learning curve shape.
3) Conduct more extensive comparisons against more recently established meta-features, especially those relating to complexities.
4) Extensive extended experimentation over more learning algorithms and more datasets, both artificial and UCI. Experimentation over more algorithms would allow us

to better facilitate algorithm selection in an Automatic Machine Learning setting. Experimentation over more datasets would provide a more robust picture of the utility of the cluster-based meta-features and would also allow us to answer the questions posed within the analysis in this paper.

5) We only explored two variants of the clustering algorithm. Other more efficient variants of this algorithm may be possible, and should be explored.
6) It may be possible to derive other meta-features from the clusters generated. This would also be a potential for further research.

References

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] I. Guyon, K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, T. K. Ho, N. Macia, B. Ray, M. Saeed, A. Statnikov *et al.*, "Design of the 2015 chalearn automl challenge," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–8.

[4] 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.

[5] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn," in *ICML workshop on AutoML*, 2014, pp. 2825–2830.

[6] J. C. Culberson, "On the futility of blind search: An algorithmic view of "no free lunch"," *Evolutionary Computation*, vol. 6, no. 2, pp. 109–127, 1998.

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

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

[9] D. W. Aha, "Generalizing from case studies: A case study," in *Machine Learning Proceedings 1992*. Elsevier, 1992, pp. 1–10.

[10] P. Brazdil, C. G. Carrier, C. Soares, and R. Vilalta, *Metalearning: Applications to data mining*. Springer Science & Business Media, 2008.

[11] S. Ali and K. A. Smith, "On learning algorithm selection for classification," *Applied Soft Computing*, vol. 6, no. 2, pp. 119–138, 2006.

[12] J. W. Lee and C. Giraud-Carrier, "Automatic selection of classification learning algorithms for data mining practitioners," *Intelligent Data Analysis*, vol. 17, no. 4, pp. 665–678, 2013.

[13] P. Brazdil and C. Giraud-Carrier, "Metalearning and algorithm selection: progress, state of the art and introduction to the 2018 special issue," 2017.

[14] K. A. Smith-Miles, "Cross-disciplinary perspectives on meta-learning for algorithm selection," *ACM Computing Surveys (CSUR)*, vol. 41, no. 1, p. 6, 2009.

[15] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial Intelligence Review*, vol. 18, no. 2, pp. 77–95, 2002.

[16] K. Alexandros and H. Melanie, "Model selection via meta-learning: a comparative study," *International Journal on Artificial Intelligence Tools*, vol. 10, no. 04, pp. 525–554, 2001.

[17] T. K. Ho and M. Basu, "Complexity measures of supervised classification problems," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 3, pp. 289–300, 2002.

[18] Y. P. P. A. F. Pavel and B. C. Soares, "Decision tree-based data characterization for meta-learning," *IDDM-2002*, p. 111, 2002.

[19] P. Brazdil, J. Gama, and B. Henery, "Characterizing the applicability of classification algorithms using meta-level learning," in *European conference on machine learning*. Springer, 1994, pp. 83–102.

[20] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017.

[21] D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, Eds., *Machine Learning, Neural and Statistical Classification*. Upper Saddle River, NJ, USA: Ellis Horwood, 1994.

[22] H. Bensusan and A. Kalousis, "Estimating the predictive accuracy of a classifier," in *European Conference on Machine Learning*. Springer, 2001, pp. 25–36.

[23] Y. Peng, P. A. Flach, C. Soares, and P. Brazdil, "Improved dataset characterisation for meta-learning," in *International Conference on Discovery Science*. Springer, 2002, pp. 141–152.

[24] B. Pfahringer, H. Bensusan, and C. G. Giraud-Carrier, "Meta-learning by landmarking various learning algorithms." in *ICML*, 2000, pp. 743–750.

[25] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Machine learning*, vol. 40, no. 3, pp. 203–228, 2000.

[26] R. Gnanadesikan, *Methods for Statistical Data Analysis of Multivariate Observations*, ser. Probability and Statistics Series. Wiley, 1977.

[27] M. Feurer, J. T. Springenberg, and F. Hutter, "Initializing bayesian hyperparameter optimization via meta-learning." in *AAAI*, 2015, pp. 1128–1135.

[28] A. C. Lorena, A. I. Maciel, P. B. de Miranda, I. G. Costa, and R. B. Prudêncio, "Data complety meta-features for regression problems," *Machine Learning*, vol. 107, no. 1, pp. 209–246, 2018.

[29] M. A. Munoz, L. Villanova, D. Baatar, and K. Smith-Miles, "Instance spaces for machine learning classification," *Machine Learning*, vol. 107, no. 1, pp. 109–147, 2018.

[30] E. T. Bell, "Exponential numbers," *The American Mathematical Monthly*, vol. 41, no. 7, pp. 411–419, 1934.

[31] D. Berend and T. Tassa, "Improved bounds on bell numbers and on moments of sums of random variables," *Probability and Mathematical Statistics*, vol. 30, no. 2, pp. 185–205, 2010.

[32] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[33] G. H. John and P. Langley, "Static versus dynamic sampling for data mining." in *KDD*, vol. 96, 1996, pp. 367–370.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.