4

Dataset Characteristics (Metafeatures)

Summary. This chapter discusses dataset characteristics that play a crucial role in many metalearning systems. Typically, they help to restrict the search in a given configuration space. The basic characteristic of the target variable, for instance, determines the choice of the right approach. If it is numeric, it suggests that a suitable regression algorithm should be used, while if it is categorical, a classification algorithm should be used instead. This chapter provides an overview of different types of dataset characteristics, which are sometimes also referred to as metafeatures. These are of different types, and include so-called simple, statistical, information-theoretic, model-based, complexity-based, and performance-based metafeatures. The last group of characteristics has the advantage that it can be easily defined in any domain. These characteristics include, for instance, *sampling landmarkers* representing the performance of particular algorithms on samples of data, *relative landmarkers* capturing differences or ratios of performance values and providing *estimates of performance gains*. The final part of this chapter discusses the specific dataset characteristics used in different machine learning tasks, including classification, regression, time series, and clustering.

4.1 Introduction

One of the goals of metalearning is to relate the performance of learning algorithms to data characteristics, i.e., *metafeatures*. Therefore, it is necessary to identify which data characteristics are good good predictors of the relative performance of algorithms and compute their values from the data. Using the framework of Rice (1976), these metafeatures can then be used to predict the performance of algorithms across datasets. This can be seen as a regression, classification, or ranking task (see Chapter 5, Sections 5.2 and 5.3).

What are good dataset features?

The development of metafeatures for metalearning should take the following issues into account:

Discriminative power. The set of metafeatures should contain information that distinguishes between the base-algorithms in terms of their performance. Therefore they should be carefully selected and represented in an adequate way.

P. Brazdil et al., *Metalearning*, Cognitive Technologies,
https://doi.org/10.1007/978-3-030-67024-5_4

Computational complexity. The metafeatures should not be too computationally complex. If this is not the case, the savings obtained by not executing all the candidate algorithms may not compensate for the cost of computing the measures used to characterize datasets. Pfahringer et al. (2000) argued that the computational complexity of metafeatures should be at most $O(n \log n)$.

Dimensionality. The number of metafeatures should not be too large compared with the amount of available metadata; otherwise overfitting may occur.

Task- and data-specific characterization

The set of metafeatures suitable for different metalearning problems may vary substantially. The best set of metafeatures for a given metalearning problem depends essentially on the task, the datasets, and the algorithms. Although this book focuses on metalearning for the recommendation of algorithms in the machine learning domain, it can be applied to various other domains. Smith-Miles (2008) discusses how metalearning can be applied to sorting, forecasting, constraint satisfaction, and optimization. Cunha et al. (2018b) discuss how metalearning can be applied to recommender systems, and Costa et al. (2020) to imbalanced domains.

Within machine learning, the most common domains are classification, regression, time series forecasting, clustering and optimization, among others. In the following sections we provide more details on the data characteristics used in some of these domains.

Characterization of algorithms

Most metalearning approaches focus on characterizing datasets. However, information about the algorithms may also be useful. For example, Hilario and Kalousis (2001) use information concerning *type of representation* (e.g., type of data they are able to deal with), *approach* (e.g., learning strategy, such as lazy or eager), *resilience* (e.g., sensitivity to irrelevant attributes, based on experimental studies), and *practicality* (e.g., easy parameter handling).

Metafeature development

Developing useful metafeatures is an essential challenge for successful metalearning systems. In metalearning, similarly to any machine learning task, this challenge is mostly addressed using (meta)feature engineering approaches, which were not very systematic initially. Recently, there is a growing interest in more systematic approaches to metafeature development.[1] We discuss this issue in more detail in Section 4.6.

4.2 Data Characterization Used in Classification Tasks

Types of Metafeatures

In this section we review the main types of features used in classification tasks. Usually, they are organized into different groups, depending on the type. Here we consider the following types:

[1] See e.g. https://ieeexplore.ieee.org/abstract/document/8215494
https://ieeexplore.ieee.org/document/7344858
https://www.ijcai.org/Proceedings/2017/0352.pdf .

1. Simple, statistical, and information-theoretic metafeatures;
2. Model-based metafeatures;
3. Performance-based metafeatures;
4. Concept and complexity metafeatures.

Each of these groups is discussed in more detail in the following subsections. Interested readers can also consult other sources which include an overview of the most common dataset features (see, e.g., Muñoz et al. (2018); Vanschoren (2019); Rivolli et al. (2019)).

4.2.1 Simple, statistical, and information-theoretic (SSI) metafeatures

Various features presented in this section represent data characteristics that are derived from dependent and/or independent variables of the given dataset.

Simple metafeatures

Typically, this set includes very simple descriptive measures, such as:

- Number of examples (instances), n;
- Number of attributes (features), p;
- Number of classes, c;
- Proportion of discrete attributes;
- Proportion of missing values of feature x_i;
- Proportion of outliers of feature x_i.

Some of these were used in the earliest metalearning approaches (e.g., Rendell et al. (1987); Aha (1992); Michie et al. (1994); Kalousis (2002)) and are still among the most commonly used metafeatures. The metafeature *number of classes* characterizes the complexity of the classification task. Some authors use different variants of some of the metafeatures shown above. For instance, instead of using number of examples, n, some researchers use $log(n)$. Some ratios of two metafeatures seem rather useful:

- Number of examples per class n/c.
- Number of examples per dimension (feature) n/p.

Normally, we would want the value of the *number of examples per class (n/c)* to be sufficiently high. It provides an estimate of data density. If the value is low, it indicates that the data is sparse, and consequently, the classification problem may be more difficult. Similarly, we would want the value of the *number of examples per dimension (n/p)* to be high too. If it is low, it indicates that we have rather too many base-level features to choose from. Michie et al. (1994) referred to this situation as the *curse of dimensionality*.

Some metafeatures refer to a particular dataset feature (e.g., proportion of outliers of feature x_i). Aggregation operations across different features are discussed in Section 4.6.

Statistical metafeatures

The most common approach to data characterization consists of the use of descriptive statistics, typically associated with numeric features.[2] Some metafeatures, such as the ones shown below, focus on a single independent feature (x_i) or a class (y).

[2]These features were used extensively in early works on metalearning (Michie et al., 1994; Brazdil et al., 1994; Brazdil and Henery, 1994; Gama and Brazdil, 1995; Todorovski and Džeroski, 1999; Lindner and Studer, 1999; Bensusan and Kalousis, 2001; Kalousis and Theoharis, 1999; Sohn, 1999; Vilalta, 1999; Köpf et al., 2000; Kalousis, 2002).

- Skewness of x_i;
- Kurtosis of x_i;
- Probability of class y.

Skewness and kurtosis characterize the shape of the underlying distribution (e.g., normality). Other metafeatures characterize the relationship between two or more independent features. These include, for instance:

- Correlation of x_i and x_j, $\rho(x_i, x_j)$;
- Covariance of x_i and x_j;
- Concentration of x_i and x_j.

The first two were discussed by Michie et al. (1994), and concentration was discussed by Kalousis and Hilario (2001b). These measures provide an estimate of feature interdependence.

The metafeatures shown in this subsection (and other subsections too) can give rise to different derived metafeatures. For instance, it is possible to apply *aggregation operations* (e.g., mean, max) to derive new metafeatures, such as *mean correlation*, from individual values. Section 4.6 discusses the details of different operations that can be used to derive new features.

Information-theoretic metafeatures

These metafeatures originated in information theory and are typically associated with nominal attributes. Some metafeatures apply to just one attribute or the class:

- Feature entropy of x_i, $H(x_i)$;
- Class entropy of y, $H(y)$.

Class entropy provides an estimate of the difficulty of the classification task (Michie et al., 1994). It can also provide an estimate of class imbalance. Other metafeatures characterize the relationship between two or more independent features:

- Mutual information between x_i and y, $MI(x_i, y)$.

Other metafeatures can be derived from the basic ones above (Michie et al., 1994):

- Intrinsic task dimensionality, $\frac{H(y)}{MI(x_i,y))}$;
- Noise-signal ratio, $\frac{H(y)-MI(x_i,y)}{MI(x_i,y)}$.

4.2.2 Model-based metafeatures

In this approach a model is induced from the data and the metafeatures are based on their properties (Bensusan, 1998; Peng et al., 2002). The model used here depends on the type of task. When dealing with classification tasks, it is possible to use, for instance, a decision tree. This type of model would obviously not be inappropriate, if we were dealing with some other ML task (e.g., regression). The model must be related in some way to the candidate algorithms to provide metafeatures that are useful. Metafeatures obtained using this approach are only useful for algorithm recommendation if the induction of the model is sufficiently fast. Some examples of some basic tree-based metafeatures reflecting concept complexity are:

- Number of nodes;

- Number of leaves;
- Branch length.

Other metafeatures can be derived from the basic ones:

- Number of nodes per feature;
- Number of leaves per class;
- Leaves agreement.

Note that, while the SSI metafeatures discussed earlier are computed directly on the dataset, model-based metafeatures are obtained indirectly through a model.

4.2.3 Performance-based metafeatures

Landmarkers

Yet another approach to data characterization is the use of *landmarkers* (Bensusan and Giraud-Carrier, 2000; Pfahringer et al., 2000).[3] Landmarkers are quick estimates of algorithm performance on a given dataset. They can be obtained by running simplified versions of the algorithms.[4] For instance, a decision stump, i.e., the root node of a decision tree, can be the landmarker for decision trees. The following landmarkers were suggested by Pfahringer et al. (2000):

- 1NN, characterizing data sparsity;
- Decision tree (or decision stump), characterizing data separability;
- Linear discriminant, characterizing linear separability;
- Naive Bayes, characterizing feature independence.

Like model-based metafeatures, landmarkers characterize the dataset indirectly. But they go one step further, by representing the performance of a model on some dataset rather than representing properties of the model.

Several studies report on comparisons of some of the approaches for data characterization discussed here (e.g., Bensusan and Kalousis (2001); Köpf and Iglezakis (2002); Todorovski et al. (2002)).

Relative landmarkers

Relative landmarkers can also be used to characterize datasets. As in the previous case, the characterization is indirect. Relative landmarkers are based on a difference (or a ratio) of the performance of two algorithms. Relative landmarkers were used for *probing the performance* of a particular algorithm a, as its performance can be compared with the performance of other algorithms (Fürnkranz and Petrak, 2001; Soares et al., 2001). Furthermore, Leite et al. (2012) used relative landmarkers in the so-called *active testing* method, discussed in Chapter 5. Finally, Post et al. (2016) used relative landmarkers to determine whether feature selection should be applied for a given algorithm and dataset combination.

[3]The concept of landmarkers can be related to earlier work on yardsticks (Brazdil et al., 1994).

[4]Chapter 3 explains how such estimates of performance can be obtained.

Subsampling landmarkers and partial learning curves

An alternative way of obtaining quick performance estimates is to run the algorithms whose performance we wish to estimate on a sample of the data, obtaining the so-called *subsampling landmarkers* (Fürnkranz and Petrak, 2001; Soares et al., 2001; Leite and Brazdil, 2004).

A more informative characteristic is obtained by considering an ordered sequence of subsampling landmarkers for a single algorithm, representing, in effect, a part of its learning curve (Leite and Brazdil, 2005). In this case, metalearning can take into account not only the values of the estimates, but also the shape of the curve.

As with the previous two cases, subsampling landmarkers also characterize the dataset indirectly. If the performance of the subsampling landmarkers were, in fact, related to the performance of the base-algorithms, one can expect this approach to be more successful than the previous ones. Experimental results exist to support this (Leite and Brazdil, 2007; van Rijn et al., 2015).

Multiple performance landmarkers

As we have pointed out in one of the previous subsections, a *landmarker* represents the performance of a particular algorithm on a particular dataset. There is no reason why we could not associate more than one landmarker with a particular dataset and represent them in the form of a vector.

4.2.4 Concept and complexity-based metafeatures

In this section we discuss a group of measures that characterize the complexity of the supervised classification task (Rendell and Seshu, 1990; Ho and Basu, 2002). Some of these measures can serve as useful metafeatures. Here we consider the following types of measures:

- Concept variation/roughness in output space;
- Overlap of individual features;
- Separability of classes.

More details about each type are given in the following subsections. Most of the metafeatures were discussed by Ho and Basu (2002), unless otherwise stated. Smith et al. (2014) use similar features, but characterize the complexity of specific instances, rather than the full dataset.

Concept variation/roughness in output space

Concept variation (Rendell and Seshu, 1990; Perez and Rendell, 1996) captures the roughness of the target concept in instance space. Irregularity in the output space occurs when neighboring examples in the input space have different labels. The measure $\delta(e_i, e_j)$ is 0 if two neighboring examples e_i and e_j belong to the same class, and 1 otherwise. The pairs of examples used differed only in one feature. The values across different pairs were then averaged to obtain the final value.

Nonlinearity of linear classifier: This measure is sensitive to the smoothness of the classifier's decision boundary, and so the aim is similar to the *concept variation* discussed earlier. The aim is to slightly alter the input points (examples), use these points as test points, and investigate the effects on the error rate of the linear classifier. The new test points are generated by repeatedly picking two points (examples) of the same class and performing a *linear interpolation* (with random coefficients) on the corresponding feature values. The classifier trained on the original training set is applied to this new test set, and its error represents this measure.

Nonlinearity of 1NN classifier: This measure is obtained in a similar way to the *nonlinearity of linear classifier*. The new test generated in the way described above is applied to the 1NN classifier trained on the original training set. The error of this classifier represents this measure.

Overlap of individual features

Fisher's discriminant ratio: This is calculated as $\frac{\mu_1 - \mu_2}{\sigma_1 - \sigma_2}$, where μ_1 and σ_1 represent the mean and standard deviation of feature values associated with class 1. Similarly, μ_2 and σ_2 are associated with class 2.

Volume overlap region: It is possible to determine a region delimited by the maximum and minimum values of some feature associated with class 1. This can be repeated also for class 2. Finally, it is possible to calculate the overlap region.

Feature efficiency: The aim is to characterize how much each feature contributes towards the separation of the two classes. If some feature values can lead to both classes, the classes are *ambiguous* in that region of values. It is possible to eliminate ambiguity progressively. In each pass, the features can be ordered by how many points are in the nonoverlapping region. The *efficiency* of each feature is defined as the fraction of all remaining points separable by that feature.

More details on the above features can be found in the article by Ho and Basu (2002).

Separability of classes

Ho and Basu (2002) proposed two groups of measures. The first one characterizes linear separability and the second whether the two sets of points (examples) come from two different distributions. Below we present just one feature from each group. Both metafeatures provide an estimate about how hard a given classification problem is.

Linear separability: This approach presupposes the application of a linear classifier. One metafeature is defined as the error rate of the linear classifier.

Fraction of points on the class boundary: The aim is to determine whether two samples (of class 1 and 2) come from the same distribution. The method uses the concept of *minimum spanning tree (MST)* to do this. The MST connects points (data examples) regardless of the class. Then the number of points connected to the opposite class represent the *points on the class boundary*. Figure 4.1 shows an illustrative example. The fraction of such points is used as one of the measures.

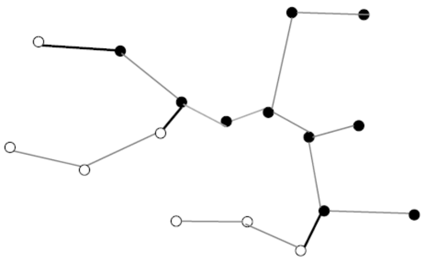

Fig. 4.1: A minimum spanning tree connecting points of two classes. The thicker edges connect two classes. Reproduced from Ho and Basu (2002)

Relationship of some complexity measures to other types

It is interesting to note that some measures discussed in this subsection presuppose a usage of a certain model type (linear classifier, NN classifier) and the measures are derived from their application. One can compare this to the model-based features discussed in Subsection 4.2.2. Also, as some features are represented by the error rates of a particular classifier (linear classifier, NN classifier), this approach could be compared to the landmarker approach discussed in Subsection 4.2.3.

4.3 Data Characterization Used in Regression Tasks

Various researchers have studied the application of metalearning approaches to regression tasks and consequently also discussed the metafeatures used (Soares et al., 2004; Lorena et al., 2018). The metafeatures can be divided into the following major groups:

- Simple and statistical metafeatures;
- Complexity-based metafeatures;
- Smoothness metafeatures.

Here we follow rather closely the exposition presented by Lorena et al. (2018), unless stated otherwise.

4.3.1 Simple and statistical metafeatures

These metafeatures are not too different from the ones discussed in Subsection 4.2.1. Many of these features can be reused, and for this reason they will not be included here. However, as the target variable is numeric, all features that involve the target variable need to be altered. Some features that characterize just one variable are:

- Coefficient of variation of the target variable, $\frac{\sigma(y)}{\mu(y)}$;
- Number of outliers of the target variable y.

The coefficient of variation of the target variable is calculated as the ratio of the standard deviation $\sigma(y)$ and the mean $\mu(y)$ of the target variable (Soares and Brazdil, 2006; Soares, 2004). Some metafeatures capture the relationship between two variables:

- Data density, n/p.

The concept of data density is similar to the concept used in classification tasks (see Subsection 4.2.1). It is calculated as the ratio of the number of examples and features.

Correlation-based metafeatures

- Correlation between feature x_i and target y, $\rho(x_i, y)$;
- Correlation between feature x_i and feature x_j, $\rho(x_i, x_j)$.

Other features can be derived from the basic set using various operations, including, e.g., aggregation operations (average, etc.), described in Section 4.6. Two metafeatures seem particularly important (Lorena et al., 2018):

- Maximum correlation between features and the target, ρ_{max};
- Average correlation between features and the target variable, $\bar{\rho}$.

A high value of ρ_{max} indicates that it may be possible to obtain good predictions of the target using this feature alone.

4.3.2 Complexity-based measures

Maximum individual feature efficiency: This measure can be seen as an adaptation of feature efficiency defined for classification tasks to regression. The concept of *high effect of feature on separability of classes* is substituted by *high effect of feature on correlation to target.* For each feature x_i the method identifies the smallest number of examples that must be removed until a high correlation (< 0.9) between feature x_i and the target variable is obtained. The numbers of examples removed are then converted to proportions. Finally, this measure is equal to the minimum proportion identified across all features. Small values indicate relatively easy problems.

Collective feature efficiency: This involves an iterative process of identifying the feature with the highest correlation, carrying out a linear fit, and eliminating examples with small residual value. This measure corresponds to the proportion of examples that remain after all features have been examined. Small values indicate relatively easy problems and higher values more complex ones.

4.3.3 Complexity/model-based measures

Lorena et al. (2018) include the following two features among the complexity-based measures. However, as they are derived from a particular model (linear regressor), we could regard them also as model-based features.

Mean absolute value of linear regressor: This measure averages the residuals of a multiple linear regression. Small values indicate simpler problems.

Variance of residuals of linear regressor: This measure averages the squares of residuals of a multiple linear regression. Small values indicate simpler problems.

4.3.4 Smoothness measures

Similarity of target values for similar examples:[5] This metafeature has a similar aim to *concept variation* in that it tries to estimate the smoothness/roughness of similar examples in the output space (see Subsection 4.2.4). The method borrows the idea of *minimum spanning tree* (MST) used in the definition of *fraction of points on the boundary*. The MST joins the most similar examples in the input (feature) space, while the edges are weighted by the Euclidean distance. This measure then captures the mean distance between the target values. Lower values indicate simpler problems.

Similarity of features for examples with similar targets:[6] This measure complements the measure above. It measures how similar the inputs (features) are for pairs of examples with similar target values.

Errors of 1NN regressor: This metafeature is an adaptation of a similar metafeature, namely the 1NN landmarker, defined for classification tasks. A suitable error measure, such as *mean squared error* (MSE), needs to be used here.

4.3.5 Nonlinearity measures

Nonlinearity of linear regressor: This metafeature is an adaptation of a similar metafeature defined for classification tasks. First, two examples with similar outputs are selected and both input (feature) and output values are interpolated to generate a new test data item. This step is repeated. The linear regressor is trained on the original data and applied to the new test set. The *mean squared error* (MSE) obtained is used as the metafeature. Lower values indicated simpler problems.

Nonlinearity of 1NN regressor: This metafeature is an adaptation of a similar metafeature defined for classification tasks. Instead of a linear regressor, the 1NN regressor is used.

4.4 Data Characterization Used in Time Series Tasks

The issue of how to apply metalearning to time series tasks was investigated by various researchers in the past (see, e.g., Adya et al. (2001); Prudêncio and Ludermir (2004); dos Santos et al. (2004); Lemke and Gabrys (2010), etc.). Characterization of time series data needs to take into account the fact that a time series is an ordered set of values.

Lemke and Gabrys (2010) divided the features into four groups: general statistics, frequency domain characteristics, autocorrelation characteristics and diversity measures. Further details about the first three groups are given in the following subsections. The last group that involves diversity measures is useful in the construction of ensembles. More details about this issue can be found in Chapter 10.

General statistics (descriptive statistics)

To calculate the descriptive statistics of the time series, Lemke and Gabrys (2010) first detrended it using polynomial regression. Some of the characteristics used are shown below:

[5]Lorena et al. (2018) call this measure the *output distribution*.

[6]Lorena et al. (2018) call this measure the *input distribution*.

- Length of time series;
- Standard deviation (std) of detrended series;
- Skewness and kurtosis;
- Trend, calculated as std(series)/std(detrended series);
- Number of turning points;
- Number of step changes;
- Estimate of nonlinearity;

The estimate of nonlinearity is obtained by generating a surrogate linear time series and comparing it with the original one.

Frequency-domain characteristics

Frequency-based features can be derived from the power spectrum which, in turn, is obtained by applying fast Fourier transform to the time series data. Lemke and Gabrys (2010), for instance, used the following features:

- Frequencies of three largest values;
- Maximal value indicating the strongest seasonal or cyclic component;
- Number of peaks that have at least 60% of the maximal component.

Autocorrelation-based characteristics

These features provide information about the stationarity and seasonability of time series. Autocorrelation and partial autocorrelation (Box and Jenkins, 2008) provide important information about the properties of a time series (Chatfield, 2003). These values are calculated with respect to data points that include a *lag* by d positions. Lemke and Gabrys (2010) used:

- Autocorrelation at lags 1 and 2;
- Partial autocorrelation at lags 1 and 2;
- Partial autocorrelation at lag 7 (or 12) capturing weakly (or monthly) seasonality.

Other metafeatures can be derived from the basic characteristics in a similar way as discussed earlier. Some examples include the *mean absolute value of the first five autocorrelations* (i.e., with $d \in \{1, \ldots, 5\}$) or the *statistical significance of the first autocorrelation coefficients* (Prudêncio and Ludermir, 2004; dos Santos et al., 2004).

4.5 Data Characterization Used in Clustering Tasks

In this section we analyze various metafeatures that can be used in clustering. This problem was addressed by various researchers before (de Souto et al., 2008; Soares et al., 2009; Ferrari and de Castro, 2015; Pimentel and de Carvalho, 2019).

This area represents a challenge, as it belongs to a group of algorithms referred to as unsupervised learning. As these tasks do not contain a target variable, fewer descriptive characteristics are available to describe the data. In the following we describe different techniques that can be used to respond to this challenge.

In Section 4.2 we presented the main types of metafeatures that tend to be used in classification tasks. They were divided into four major groups. So a question arises regarding whether each group can be adapted to clustering tasks and, if so, how. The following subsections provide details on this topic.

Simple, statistical, and information-theoretic metafeatures

As the data does not include the target variable, it is possible to use only the features that involve independent variables. Ferrari and de Castro (2015), for instance, used an appropriate subset of metafeatures, similar to those shown in Subsection 4.2.1. Pimentel and de Carvalho (2019) proposed metafeatures describing the distribution of rank correlation between examples (not features).

Model-based metafeatures

Interestingly, Ferrari and de Castro (2015) adapted this idea to the task of clustering. The authors defined a vector **d** containing pairwise Euclidean distances $d_{i,j}$ between all pairs of objects (data instances) i and j of a given dataset. The vector is then normalized into an interval $[0, 1]$ and characterized using statistical measures. These can be divided into three groups discussed next.

The first subgroup includes some simple measures, such as mean, variance, standard deviation, skewness, and kurtosis. All these characterize the distribution of values in **d**.

The second subgroup of metafeatures characterizes a histogram constructed on the basis of distribution of the values in **d**, following the approach of Kalousis (2002). The authors used 10 bins (intervals) of equal size. The feature corresponding to bin j includes the percentage of values contained in this bin.

The third subgroup of metafeatures provides an alternative way of characterizing the distribution. The authors have first generated z-scores defined by $z = \frac{x-\mu}{\sigma}$, where μ represents the mean and σ standard deviation. The absolute values of z scores were discretized into four bins: $[0, 1), [1, 2), [2, 3)$, and $[3, \infty)$. The corresponding metafeatures captured the proportion of cases in each bin.

Performance-based metafeatures

Some authors have used *internal validation measures* in the meta-learning framework for clustering (Vukicevic et al., 2016; Tomp et al., 2019).

It would seem that various measures that were used in classification tasks, such as landmarkers and subsampling landmarkers, could be adapted to this domain.

Metalearning vs. optimization on target dataset

This domain provides, however, a challenge to metalearning approaches. It may be difficult to provide a good recommendation of clustering algorithms (or their configurations) just by looking at the data. This is because many approaches do not cluster the points in their original space, but rather use dimensionality reduction first. So, the alternative is to carry out a search for the best solution on the target concept.

4.6 Deriving New Features from the Basic Set

Generating new features by aggregation

We note that some features, such as *skewness*, can be calculated for each numeric attribute. Given that the number of attributes varies for different datasets, this implies that the number of values describing *skewness* for different datasets varies. This creates a problem for metalearning systems that use propositional representation.

The most common approach to solve this problem is to do some form of aggregation, for instance by calculating *mean skewness*. However, it should be expected that important information may be lost by this aggregation. Alternatively, Kalousis and Theoharis (1999) used a finer-grained aggregation, where histograms with a fixed number of bins were used to construct new metafeatures. For instance, the distribution of skewness values could be represented with three metafeatures corresponding to the number of attributes with skewness smaller than 0.2, between 0.2 and 0.4, and larger than 0.4.

Generating a complete set of metafeatures

Some researchers (Pinto et al., 2016; Pinto, 2018) have observed that many systems use a set of dataset features that can be considered incomplete. For instance, *entropy* is commonly applied to the target variable, but not to dataset features. Aggregation operations often involve calculating, for instance, the mean value of all numeric features. Different aggregation operations (see, e.g., Tukey (1977)) are listed below:

- mean value (μ)
- standard deviation (σ)
- minimum value (min)
- maximum value (max)
- first quartile (q1)
- median value (q2)
- third quartile (q3)

These are often not used. So the authors have proposed to generate a complete set of features, thus enriching the initial set that was considered. Pinto (2018) has shown that a metalearning system that uses the complete set achieves better performance than the initial set. Although feature selection can be used to reduce this set, it was shown that it could degrade performance.

Generating new features by PCA

Principal component analysis can be used to project features into a low-dimensional space that includes the principal components. This method was used, for instance, by Smith-Miles et al. (2014). However, the PCA model was somewhat unsatisfactory to predict performance, since PCA is only concerned with maximizing the variance explained by the features.

Transforming features by feature selection and projection

The method used by Smith-Miles et al. (2014), which included 235 dataset instances, used two steps. In the first one, feature selection was used to reduce a relatively large set of features (509) to ten features. In the second step, the ten-dimensional space was projected onto a two-dimensional space. The projection was defined as an optimization problem, where the aim was to minimize the *approximation error*, defined in terms of both true and predicted values of the feature data matrix and performance vector. The projection revealed regions in the 2D space, referred to as a *footprint*, where a particular algorithm is expected to do well. More details about this study can be found in Subsection 4.7.1.

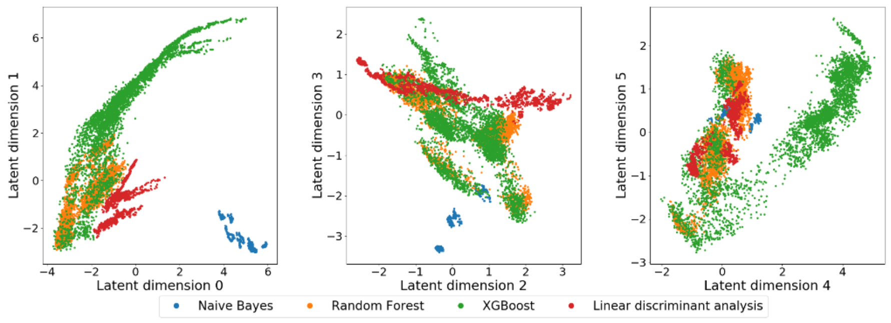

Fig. 4.2: Latent embedding based on probabilistic matrix decomposition of $42{,}000$ configurations, color coded by the algorithm. Image taken from Fusi et al. (2018)

Constructing new latent features by matrix factorization

Fusi et al. (2018) proposed to apply matrix factorization to the performance matrix $\boldsymbol{Y} \in \mathbb{R}^{N \times D}$, where N is the number of algorithms (workflows) and D is the number of datasets. Each cell contains the performance of a particular algorithm on a particular dataset. The authors proposed to use a probabilistic matrix factorization algorithm that decomposes $\boldsymbol{Y} \approx \boldsymbol{X}\boldsymbol{W}$, where $\boldsymbol{X} \in \mathbb{R}^{N \times Q}$ and $\boldsymbol{W} \in \mathbb{R}^{Q \times D}$.

The authors note that, in many cases, $\boldsymbol{Y}$ is a sparse matrix and this method helps to provide a solution. Being able to deal with missing values is important in situations when a repeated algorithm selection method is carried out for the new (target) dataset. An advantage of the probabilistic matrix factorization method is that it maps each dataset into a latent feature vector of size Q.

This opens up a way to new lines of research, where the resulting matrix can be mined for patterns, such as shown in Figure 4.2. The plot shows various results on several datasets relative to four different algorithms (more precisely OpenML flows) and their differently configured variants. The algorithms involved in this study were naive Bayes, random forest, XGBoost, and linear discriminant analysis (LDA).

Yang et al. (2019) describe an AutoML system that is integrated with an algorithm selection method with similar latent features.

Generating new features in the form of embeddings

Some researchers have suggested to use a so-called Siamese neural network (SNN) to generate a feature vector from a given dataset (Baldi and Chauvin, 1993; Bromley et al., 1994).[7] This network consists of two similar sub-networks. During training each sub-network is applied to similar examples, such as, examples from one class. This permits to extract a feature vector consisting of neural weights, representing effectively an embedding. Classification consists of comparing an extracted feature vector of each example

[7]Chapter 13 discusses Siamese neural networks.

with a stored feature vector for each class. Items closer to this stored representation for the positive class than a chosen threshold are accepted as belonging to the positive class.

This approach was originally used to distinguish original signatures from forgeries. In subsequent works it was reused and adapted to various other domains, including, e.g., speaker recognition (Chen and Salman, 2011) and sentence similarity (Mueller and Thyagarajan, 2016).

In another work on recommender systems which involves a metalearning approach, Cunha et al. (2018a) used graph embeddings to create *dataset embeddings*.

4.7 Selection of Metafeatures

4.7.1 Static selection of metafeatures

It is often important to select a suitable subset of data characteristics and the corresponding metafeatures from all the possible alternatives. The number of metafeatures should not be too large compared with the amount of available metadata. An excessively large number of measures may cause overfitting and, thus, poor predictions on unseen data. This is particularly true because the number of examples in metalearning (i.e., datasets) tends to be small.

Selection of metafeatures may be done during the development of the metalearning system by including only measures that are expected to be relevant (Brazdil et al., 2003). This can be done by taking into account the characteristics of the metalearning problem, as discussed above.

Alternatively, it is possible to include as many metafeatures as possible. A feature selection method can then be applied to obtain a smaller subset of suitable metafeatures. Obviously, if it were possible to use a relatively small subset, this would have advantages. The whole process of metalearning would be simpler, as one would not have to calculate so many characteristics.

It has been shown that the use of wrapper-based feature selection methods at the meta level can improve the quality of the results (Todorovski et al., 2000; Kalousis and Hilario, 2001a). The improvement can be attributed to the fact that "noisy" attributes have been dropped. The wrapper-based approach normally uses *backward elimination*, which is normally used for feature selection (Kuhn and Johnson, 2013).

Recently, Muñoz et al. (2018) carried out a comprehensive study in the area of classification with 509 features. The aim was to select a small subset that would characterize well the hardness of the classification task. The level of hardness was established by measures, such as *nonlinear separability*, among others. The whole process of identifying the relevant features is quite complex, so the authors have identified the following ten features:

- Maximum normalized entropy of the attributes (information-theoretic)
- Normalized entropy of class attribute (information-theoretic)
- Mean mutual information of attributes and class (information-theoretic)
- Error rate of the decision node (landmarker)
- Training error of linear classifier (landmarker)
- Standard deviation of the weighted distance (concept characterization)
- Maximum feature efficiency (complexity)
- Collective feature efficiency (complexity)
- Fraction of points on the class boundary (complexity)

- Nonlinearity of nearest-neighbor classifier (complexity)

The type of metafeature is mentioned as well. We note that this set includes representatives of several metafeature types.

4.7.2 Dynamic (iterative) data characterization

In the previous subsection we described the process of selecting metafeatures prior to their use by a metalearning system. An alternative approach consists of gathering the metafeatures in an iterative fashion (Leite and Brazdil, 2005, 2007). This approach is useful in situations when gathering the metafeatures incurs costs. We may not want to use the most informative set from the start simply to save effort.

Suppose the aim is to determine whether algorithm A or algorithm B should be used with the target dataset. A test of both algorithms on a small sample (i.e., on the basis of a given *subsampling landmarker*) provides some information that can be used for this decision. Obviously, if we use more samples, we obtain more information. But if we can make a decision on the basis of the existing metafeatures, there is no need to extend it further.

In each phase of the algorithm described by Leite and Brazdil (2005, 2007), the system tries to determine whether the currently available set of metafeatures is adequate or whether it should be extended, and if so, how. This is done with the help of existing metadata. The aim is to determine what happened in similar circumstances in the past. If there is evidence that some extensions lead to a marked improvement of performance, the system tries to identify the best one. This is the one which is expected to provide maximum information while requiring the least computational effort.

We note that characterization of datasets is built up gradually. In each step, the system determines the next sample sizes that should be tried out. The plan of these experiments is built up gradually, by taking into account the results of all previous experiments, both on other datasets (past metadata) and partly also on the target dataset (new metadata).

4.8 Algorithm-Specific Characterization and Representation Issues

4.8.1 Algorithm-specific data characterization

The set of base-algorithms should also be taken into account in the development of metafeatures. In the case where diverse algorithms are included, different sets of metafeatures could be useful for discriminating the performance of different pairs of algorithms (Aha, 1992; Kalousis and Hilario, 2001a, 2000; Sun and Pfahringer, 2013).

For instance, the *proportion of continuous features* can be useful to discriminate between naive Bayes and k-NN, but not between naive Bayes and a rule-based learner (Kalousis and Hilario, 2000). This is consistent with the knowledge that k-NN is better suited for continuous features than naive Bayes, but both the naive Bayes and rule-based systems have problems to deal with this kind of attributes. Therefore, a set of metafeatures that is able to discriminate among all of the algorithms should be used.

Data characterization useful for ranking pairs of algorithms

Another approach is to transform the problem into several pairwise metalearning problems (i.e., predict whether to use algorithm A or B, or whether they are equivalent) and use different sets of metafeatures for each of them. This strategy has been used, for instance, by Sun and Pfahringer (2013). The existing meta-data is used to train a rule-based classifier whose aim is to predict whether algorithm A (or B) is better for a particular dataset. The rules include base-level features which may include landmarkers (e.g., AUC associated with a particular type of tree (REPTree.depth2)).

When the base-algorithms are similar, specific metafeatures that represent the differences between them should be designed. A particular case is when the base-algorithms represent the same algorithm with different parameter settings. In the case of selecting parameters for the kernel of SVM, it has been shown that better results are obtained with algorithm-specific metafeatures than with general ones (Soares and Brazdil, 2006). The metafeatures used in this work were based on the kernel matrices for the different kernel parameters considered. In a different approach to this problem, metafeatures characterizing the kernel matrix were combined with other metafeatures describing the data in terms of its relation to the margin (Tsuda et al., 2001).

4.8.2 Representation Issues

Most researchers represent the metafeatures using a vector with a fixed number of positions. However, some approaches have exploited a *relational representation* of metafeatures (Todorovski and Džeroski, 1999; Hilario and Kalousis, 2001; Kalousis and Hilario, 2003), commonly used in inductive logic programming (ILP). For instance, in a dataset with k_c continuous attributes, skewness is described by k_c metafeatures, with the skewness value of each attribute. An ILP approach has also been proposed to take full advantage of the model-based approach to data characterization, which is also non-propositional (Bensusan et al., 2000). The authors illustrate their proposal by characterizing the dataset using a decision tree induced from that dataset.

4.9 Establishing Similarity Between Datasets

4.9.1 Similarity based on metafeatures

Let us represent the metafeatures of some meta-instance (dataset) d_i using a vector $\mathsf{f}_{d_i} = (\mathsf{f}_{d_i,1}, \mathsf{f}_{d_i,2}, \cdots, \mathsf{f}_{d_i,m})$, where m is the number of metafeatures. Similarly, $\mathsf{f}_{d_{new}}$ represents the vector of metafeatures of the target dataset d_{new}. The vectors of metafeatures are used to identify the datasets that are most similar to the target dataset. This is done using an approach similar to *k*-NN. The similarity between examples is usually based on some simple distance measure (e.g., Manhattan, Euclidean, etc.).[8]

The following formula shows how we can calculate the distance between dataset d_{new} and dataset d_i, assuming that all features are numeric and are attributed equal weight:

$$Dist_{mf}(d_{new}, d_i) = \sum_{p=1}^{\mathsf{m}} \frac{|\mathsf{f}_{d_{new},p} - \mathsf{f}_{d_i,p}|}{\max(\mathsf{f}_{*,p}) - \min(\mathsf{f}_{*,p})}. \tag{4.1}$$

[8]Distance measures are discussed by Atkeson et al. (1997).

This formula uses the *L1 norm* in the calculation of the distance. The distance value for each metafeature is normalized by dividing it by the corresponding range of values across all datasets. The value of similarity based on metafeatures can be obtained from the distance using: $Sim_{mf} = 1 - Dist_{mf}$.

4.9.2 Similarity based on performance results of algorithms

The following two similarity measures are based on recent work of Leite and Brazdil (2021).

Cosine-based similarity of performance results

This version of similarity calculates the similarity between two datasets by considering the performance results of different algorithms on these datasets. The actual measure used here is *cosine similarity*. This measure permits to calculate the similarity between two vectors $\mathbf{v}(d_{new})$ and $\mathbf{v}(d_i)$ representing dataset d_{new} and d_i as follows (Manning et al., 2009):

$$Sim_{cos}(d_{new}, d_i) = \frac{\mathbf{v}(d_{new}) \cdot \mathbf{v}(d_i)}{|\mathbf{v}(d_{new})|_2 * |\mathbf{v}(d_i)|_2}, \tag{4.2}$$

where the numerator represents the *dot product* (inner product) of the two vectors, while the denominator is the product of their Euclidean lengths. The denominator is used to normalize the resulting values so that they would be in the range between 0 and 1. Here the vectors $\mathbf{v}(d_{new})$ and $\mathbf{v}(d_i)$ represent performance values obtained by evaluating algorithms (workflows) **a** on dataset d_{new} or d_i respectively. This can be represented by function $p(\mathbf{a}, d_{new})$ and $p(\mathbf{a}, d_i)$. Consequently, the equation above can be rewritten as

$$Sim_{cos}(d_{new}, d_i) = \frac{p(\mathbf{a}, d_{new}) \cdot p(\mathbf{a}, d_i)}{|p(\mathbf{a}, d_{new})|_2 * |p(\mathbf{a}, d_i)|_2}. \tag{4.3}$$

After substituting the dot product by the sum of products we get

$$Sim_{cos}(d_{new}, d_i) = \frac{\sum_{a_k \in \mathbf{a}} p(a_k, d_i) * p(a_k, d_{new})}{|p(\mathbf{a}, d_{new})|_2 * |p(\mathbf{a}, d_i)|_2}. \tag{4.4}$$

The Euclidean length of the terms in the denominator of the form $|\mathbf{x}|_2$ is calculated as $\sum \sqrt{x_k})^2$. The algorithm set **a** is a subset of all possible elements which were already evaluated on d_{new}.

Correlation-based similarity of performance results

This version of similarity is similar to the previous one. It also calculates similarity between two datasets by considering the performance results of different algorithms on these datasets. Instead of using cosine similarity it uses similarity based on Spearman's correlation

$$Sim_{rs}(d_{new}, d_i) = r_s(p(\mathbf{a}, d_{new}), p(\mathbf{a}, d_i)), \tag{4.5}$$

where $p(\mathbf{a}, d_{new})$ is a function applied to a vector of algorithms (workflows) **a** that returns the corresponding performance values (e.g., accuracies) on dataset d_{new}, and $p(\mathbf{a}, d_i)$ is defined in a similar way. The term r_s represents Spearman's correlation function.

Another similar variant uses a weighted rank measure of correlation (da Costa and Soares, 2005; da Costa, 2015) discussed in Chapter 3 (Section 3.2)

$$Sim_{rw}(d_{new}, d_i) = r_w(p(\mathbf{a}, d_{new}), p(\mathbf{a}, d_i)). \tag{4.6}$$

Similarly, r_w represents the weighted rank correlation function.

References

Adya, M., Collopy, F., Armstrong, J., and Kennedy, M. (2001). Automatic identification of time series features for rule-based forecasting. *International Journal of Forecasting*, 17(2):143–157.

Aha, D. W. (1992). Generalizing from case studies: A case study. In Sleeman, D. and Edwards, P., editors, *Proceedings of the Ninth International Workshop on Machine Learning (ML92)*, pages 1–10. Morgan Kaufmann.

Atkeson, C. G., Moore, A. W., and Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73.

Baldi, P. and Chauvin, Y. (1993). Neural networks for fingerprint recognition. *Neural Computation*, 5.

Bensusan, H. (1998). God doesn't always shave with Occam's razor - learning when and how to prune. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, pages 119–124, London, UK. Springer-Verlag.

Bensusan, H. and Giraud-Carrier, C. (2000). Discovering task neighbourhoods through landmark learning performances. In Zighed, D. A., Komorowski, J., and Zytkow, J., editors, *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2000)*, pages 325–330. Springer.

Bensusan, H., Giraud-Carrier, C., and Kennedy, C. (2000). A higher-order approach to meta-learning. In *Proceedings of the ECML 2000 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, pages 109–117. ECML 2000.

Bensusan, H. and Kalousis, A. (2001). Estimating the predictive accuracy of a classifier. In Flach, P. and De Raedt, L., editors, *Proceedings of the 12th European Conference on Machine Learning*, pages 25–36. Springer.

Box, G. and Jenkins, G. (2008). *Time Series Analysis, Forecasting and Control*. John Wiley & Sons.

Brazdil, P., Gama, J., and Henery, B. (1994). Characterizing the applicability of classification algorithms using meta-level learning. In Bergadano, F. and De Raedt, L., editors, *Proceedings of the European Conference on Machine Learning (ECML94)*, pages 83–102. Springer-Verlag.

Brazdil, P. and Henery, R. J. (1994). Analysis of results. In Michie, D., Spiegelhalter, D. J., and Taylor, C. C., editors, *Machine Learning, Neural and Statistical Classification*, chapter 10, pages 175–212. Ellis Horwood.

Brazdil, P., Soares, C., and da Costa, J. P. (2003). Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning*, 50(3):251–277.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems 7*, NIPS'94, pages 737–744.

Chatfield, C. (2003). *The Analysis of Time Series: An Introduction*. Chapman & Hall/CRC, 6th edition.

Chen, K. and Salman, A. (2011). Extracting speaker-specific information with a regularized Siamese deep network. In *Advances in Neural Information Processing Systems 24*, NIPS'11, pages 298–306.

Costa, A. J., Santos, M. S., Soares, C., and Abreu, P. H. (2020). Analysis of imbalance strategies recommendation using a meta-learning approach. In *7th ICML Workshop on Automated Machine Learning (AutoML)*.

Cunha, T., Soares, C., and de Carvalho, A. C. (2018a). cf2vec: Collaborative filtering algorithm selection using graph distributed representations. *arXiv preprint arXiv:1809.06120*.

Cunha, T., Soares, C., and de Carvalho, A. C. (2018b). Metalearning and recommender systems: A literature review and empirical study on the algorithm selection problem for collaborative filtering. *Information Sciences*, 423:128 – 144.

da Costa, J. P. (2015). *Rankings and Preferences: New Results in Weighted Correlation and Weighted Principal Component Analysis with Applications*. Springer.

da Costa, J. P. and Soares, C. (2005). A weighted rank measure of correlation. *Aust. N.Z. J. Stat.*, 47(4):515–529.

de Souto, M. C. P., Prudencio, R. B. C., Soares, R. G. F., de Araujo, D. S. A., Costa, I. G., Ludermir, T. B., and Schliep, A. (2008). Ranking and selecting clustering algorithms using a meta-learning approach. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 3729–3735.

dos Santos, P. M., Ludermir, T. B., and Prudêncio, R. B. C. (2004). Selection of time series forecasting models based on performance information. In *Proceedings of the Fourth International Conference on Hybrid Intelligent Systems (HIS'04)*, pages 366–371.

Ferrari, D. and de Castro, L. (2015). Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods. *Information Sciences*, 301:181–194.

Fürnkranz, J. and Petrak, J. (2001). An evaluation of landmarking variants. In Giraud-Carrier, C., Lavrač, N., and Moyle, S., editors, *Working Notes of the ECML/PKDD 2000 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, pages 57–68.

Fusi, N., Sheth, R., and Elibol, M. (2018). Probabilistic matrix factorization for automated machine learning. In *Advances in Neural Information Processing Systems 32*, NIPS'18, pages 3348–3357.

Gama, J. and Brazdil, P. (1995). Characterization of classification algorithms. In Pinto-Ferreira, C. and Mamede, N. J., editors, *Progress in Artificial Intelligence, Proceedings of the Seventh Portuguese Conference on Artificial Intelligence*, pages 189–200. Springer-Verlag.

Hilario, M. and Kalousis, A. (2001). Fusion of meta-knowledge and meta-data for case-based model selection. In Siebes, A. and De Raedt, L., editors, *Proceedings of the Fifth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD01)*. Springer.

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

Kalousis, A. (2002). *Algorithm Selection via Meta-Learning*. PhD thesis, University of Geneva, Department of Computer Science.

Kalousis, A. and Hilario, M. (2000). Model selection via meta-learning: A comparative study. In *Proceedings of the 12th International IEEE Conference on Tools with AI*. IEEE Press.

Kalousis, A. and Hilario, M. (2001a). Feature selection for meta-learning. In Cheung, D. W., Williams, G., and Li, Q., editors, *Proc. of the Fifth Pacific-Asia Conf. on Knowledge Discovery and Data Mining*. Springer.

Kalousis, A. and Hilario, M. (2001b). Model selection via meta-learning: a comparative study. *Int. Journal on Artificial Intelligence Tools*, 10(4):525–554.

Kalousis, A. and Hilario, M. (2003). Representational issues in meta-learning. In *Proceedings of the 20th International Conference on Machine Learning*, ICML'03, pages 313–320.

Kalousis, A. and Theoharis, T. (1999). NOEMON: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis*, 3(5):319–337.

Köpf, C. and Iglezakis, I. (2002). Combination of task description strategies and case base properties for meta-learning. In Bohanec, M., Kavšek, B., Lavrač, N., and Mladenić, D., editors, *Proceedings of the Second International Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning (IDDM-2002)*, pages 65–76. Helsinki University Printing House.

Köpf, C., Taylor, C., and Keller, J. (2000). Meta-analysis: From data characterization for meta-learning to meta-regression. In Brazdil, P. and Jorge, A., 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.

Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling*. Springer.

Leite, R. and Brazdil, P. (2004). Improving progressive sampling via meta-learning on learning curves. In Boulicaut, J.-F., Esposito, F., Giannotti, F., and Pedreschi, D., editors, *Proc. of the 15th European Conf. on Machine Learning (ECML2004)*, LNAI 3201, pages 250–261. Springer-Verlag.

Leite, R. and Brazdil, P. (2005). Predicting relative performance of classifiers from samples. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML'05, pages 497–503, NY, USA. ACM Press.

Leite, R. and Brazdil, P. (2007). An iterative process for building learning curves and predicting relative performance of classifiers. In *Proceedings of the 13th Portuguese Conference on Artificial Intelligence (EPIA 2007)*, pages 87–98.

Leite, R. and Brazdil, P. (2021). Exploiting performance-based similarity between datasets in metalearning. In Guyon, I., van Rijn, J. N., Treguer, S., and Vanschoren, J., editors, *AAAI Workshop on Meta-Learning and MetaDL Challenge*, volume 140, pages 90–99. PMLR.

Leite, R., Brazdil, P., and Vanschoren, J. (2012). Selecting classification algorithms with active testing. In *Machine Learning and Data Mining in Pattern Recognition*, pages 117–131. Springer.

Lemke, C. and Gabrys, B. (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, 74:2006–2016.

Lindner, G. and Studer, R. (1999). AST: Support for algorithm selection with a CBR approach. In Giraud-Carrier, C. and Pfahringer, B., editors, *Recent Advances in Meta-Learning and Future Work*, pages 38–47. J. Stefan Institute.

Lorena, A., Maciel, A., de Miranda, P., Costa, I., and Prudêncio, R. (2018). Data complexity meta-features for regression tasks. *Machine Learning*, 107(1):209–246.

Manning, C., Raghavan, P., and Schütze, H. (2009). *An Introduction to Information Retrieval*. Cambridge University Press.

Michie, D., Spiegelhalter, D. J., and Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

Muñoz, M., Villanova, L., Baatar, D., and Smith-Miles, K. (2018). Instance Spaces for Machine Learning Classification. *Machine Learning*, 107(1).

Mueller, J. and Thyagarajan, A. (2016). Siamese recurrent architectures for learning sentence similarity. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Peng, Y., Flach, P., Brazdil, P., and Soares, C. (2002). Improved dataset characterisation for meta-learning. In *Discovery Science*, pages 141–152.

Perez, E. and Rendell, L. (1996). Learning despite concept variation by finding structure in attribute-based data. In *Proceedings of the 13th International Conference on Machine Learning*, ICML'96.

Pfahringer, B., Bensusan, H., and Giraud-Carrier, C. (2000). Meta-learning by landmarking various learning algorithms. In Langley, P., editor, *Proceedings of the 17th International Conference on Machine Learning*, ICML'00, pages 743–750.

Pimentel, B. A. and de Carvalho, A. C. (2019). A new data characterization for selecting clustering algorithms using meta-learning. *Information Sciences*, 477:203 – 219.

Pinto, F. (2018). *Leveraging Bagging for Bagging Classifiers*. PhD thesis, University of Porto, FEUP.

Pinto, F., Soares, C., and Mendes-Moreira, J. (2016). Towards automatic generation of metafeatures. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 215–226. Springer International Publishing.

Post, M. J., van der Putten, P., and van Rijn, J. N. (2016). Does feature selection improve classification? a large scale experiment in OpenML. In *Advances in Intelligent Data Analysis XV*, pages 158–170. Springer.

Prudêncio, R. and Ludermir, T. (2004). Meta-learning approaches to selecting time series models. *Neurocomputing*, 61:121–137.

Rendell, L. and Seshu, R. (1990). Learning hard concepts through constructive induction: Framework and rationale. *Computational Intelligence*, 6:247–270.

Rendell, L., Seshu, R., and Tcheng, D. (1987). More robust concept learning using dynamically-variable bias. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 66–78. Morgan Kaufmann Publishers, Inc.

Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers*, 15:65–118.

Rivolli, A., Garcia, L. P. F., Soares, C., Vanschoren, J., and de Carvalho, A. C. P. L. F. (2019). Characterizing classification datasets: a study of meta-features for meta-learning. In *arXiv*. https://arxiv.org/abs/1808.10406.

Smith, M. R., Martinez, T., and Giraud-Carrier, C. (2014). An instance level analysis of data complexity. *Machine Learning*, 95(2):225–256.

Smith-Miles, K., Baatar, D., Wreford, B., and Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, 45:12–24.

Smith-Miles, K. A. (2008). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, 41(1):6:1–6:25.

Soares, C. (2004). *Learning Rankings of Learning Algorithms*. PhD thesis, Department of Computer Science, Faculty of Sciences, University of Porto.

Soares, C. and Brazdil, P. (2006). Selecting parameters of SVM using meta-learning and kernel matrix-based meta-features. In *Proceedings of the ACM SAC*.

Soares, C., Brazdil, P., and Kuba, P. (2004). A meta-learning method to select the kernel width in support vector regression. *Machine Learning*, 54:195–209.

Soares, C., Petrak, J., and Brazdil, P. (2001). Sampling-based relative landmarks: Systematically test-driving algorithms before choosing. In Brazdil, P. and Jorge, A., editors, *Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA2001)*, pages 88–94. Springer.

Soares, R. G. F., Ludermir, T. B., and De Carvalho, F. A. T. (2009). An analysis of meta-learning techniques for ranking clustering algorithms applied to artificial data. In Alippi, C., Polycarpou, M., Panayiotou, C., and Ellinas, G., editors, *Artificial Neural Networks – ICANN 2009*. Springer, Berlin, Heidelberg.

Sohn, S. Y. (1999). Meta analysis of classification algorithms for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1137–1144.

Sun, Q. and Pfahringer, B. (2013). Pairwise meta-rules for better meta-learning-based algorithm ranking. *Machine Learning*, 93(1):141–161.

Todorovski, L., Blockeel, H., and Džeroski, S. (2002). Ranking with predictive clustering trees. In Elomaa, T., Mannila, H., and Toivonen, H., editors, *Proc. of the 13th European Conf. on Machine Learning*, number 2430 in LNAI, pages 444–455. Springer-Verlag.

Todorovski, L., Brazdil, P., and Soares, C. (2000). Report on the experiments with feature selection in meta-level learning. In Brazdil, P. and Jorge, A., editors, *Proceedings of the Data Mining, Decision Support, Meta-Learning and ILP Workshop at PKDD 2000*, pages 27–39.

Todorovski, L. and Džeroski, S. (1999). Experiments in meta-level learning with ILP. In Rauch, J. and Zytkow, J., editors, *Proceedings of the Third European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD99)*, pages 98–106. Springer.

Tomp, D., Muravyov, S., Filchenkov, A., and Parfenov, V. (2019). Meta-learning based evolutionary clustering algorithm. In *Lecture Notes in Computer Science, Vol. 11871*, pages 502–513.

Tsuda, K., Rätsch, G., Mika, S., and Müller, K. (2001). Learning to predict the leave-one-out error of kernel based classifiers. In *ICANN*, pages 331–338. Springer-Verlag.

Tukey, J. (1977). *Exploratory Data Analysis*. Addison-Wesley Publishing Company.

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

Vanschoren, J. (2019). Meta-learning. In Hutter, F., Kotthoff, L., and Vanschoren, J., editors, *Automated Machine Learning: Methods, Systems, Challenges*, chapter 2, pages 39–68. Springer.

Vilalta, R. (1999). Understanding accuracy performance through concept characterization and algorithm analysis. In Giraud-Carrier, C. and Pfahringer, B., editors, *Recent Advances in Meta-Learning and Future Work*, pages 3–9. J. Stefan Institute.

Vukicevic, M., Radovanovic, S., Delibasic, B., and Suknovic, M. (2016). Extending meta-learning framework for clustering gene expression data with component-based algorithm design and internal evaluation measures. *International Journal of Data Mining and Bioinformatics (IJDMB)*, 14(2).

Yang, C., Akimoto, Y., Kim, D. W., and Udell, M. (2019). Oboe: Collaborative filtering for AutoML model selection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1173–1183. ACM.