ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200211

Family and Prejudice: A Behavioural Taxonomy of Machine Learning Techniques

Raül Fabra-Boluda[1] and **Cèsar Ferri**[1] and **Fernando Martínez-Plumed**[1,2] and **José Hernández-Orallo**[1] and **M. José Ramírez-Quintana**[1]

Abstract. One classical way of characterising the rich range of machine learning techniques is by defining 'families', according to their formulation and learning strategy (e.g., neural networks, Bayesian methods, etc.). However, this taxonomy of learning techniques does not consider the extent to which models built with techniques from the same or different family agree on their outputs, especially when their predictions have to extrapolate in sparse zones where insufficient training data was available. In this paper we present a new taxonomy of machine learning techniques for classification, where families are clustered according to their degree of (dis)agreement in behaviour considering both dense and sparse zones, using Cohen's kappa statistic. To this end, we use a representative collection of datasets and learning techniques. We finally validate the taxonomy by performing a number of experiments for technique selection. We show that ranking techniques by only following prejudice –the reputation they have for other problems– is worse than selecting techniques based on family diversity.

1 INTRODUCTION

"It is a truth universally acknowledged, that a single ML technique in possession of a good reputation, must be in want of a dataset"[3]. Matching ML techniques with datasets is an immemorial question, as it is well-acknowledged that no technique is optimal for all problems[4]. Consequently, using the algorithm with best reputation for all problems is a major mistake: there is no best-for-all algorithm. As a reaction to this, areas such as meta-learning [3] and Auto-ML [13, 20] are like modern matchmakers that look for the most appropriate marriage, according to the characteristics of the technique and the problem at hand.

This can exploit the current landscape of techniques and hyperparameters through the use of sophisticated pipelines and millions of training episodes to find the good match. However, many practitioners usually do matchmaking in the old-fashioned way: they fall back on a few representative algorithms, or *families*, looking for diversity. This intuition is also supported by the same no-best-for-all rationale, but seen in the opposite direction. It is well-known that *different* types of learning models are good at modelling *different* kinds of underlying patterns in data [4]. If an algorithm has performed poorly for a problem, a very similar algorithm will perform poorly too. Therefore, using and testing out a diverse set of models seems a good and simple strategy that many ML practitioners and researchers do. To this purpose, machine learning algorithms are often (intuitively) grouped by similarity in terms of their formulation and learning strategy, such as decision trees, kernel methods or neural networks. This sort of grouping is what we usually find in machine learning and AI books [16, 14], in the literature [12], or in other sources such as CRAN[5], Scikit-Learn[6] or Wikipedia[7]. These taxonomies are then regularly used by machine learning users and data scientists to look for this diversity of approaches. Analogously, when a new learning algorithm is defined, the researchers use the taxonomies or families for comparing the new proposed technique with other algorithms in the same or different category, depending on the point they want to make. More recently, many adversarial attacks or explainable ML [15] approaches are classified according to those that require or benefit from knowing the model family and those that are fully blackbox.

As a result, the notion of machine learning *family* is ubiquitous, and shapes the practice and theory of the field. Even when metalearning and Auto-ML are used, there is usually a pool of initial techniques that are also based on these taxonomies. In this paper, we argue that these *family taxonomies are folklore*, and not really based on empirical or theoretical results about their similar or dissimilar behaviour. As a response, we derive a taxonomy of machine learning families for classification that is based on their behaviour, which can be used in all the facets of machine learning theory and practice, instead of the traditional folk taxonomies.

Our methodology is based on an important insight. While in dense areas differences between models may be difficult to find, in sparse areas the algorithms diverge significantly, and unveil the characteristic behaviour of the trained models using those techniques. More precisely, in classification, models generated using particular machine learning techniques may disagree not only in the difficult areas close to the decision boundaries, but also on how they extrapolate on areas with little or no training examples. Figure 1 illustrates this, where on the top-left plot we see the original training data of a bivariate dataset that we use to train several ML models (using different techniques). What we observe is that all these models behave similarly (i.e., produce similar partitions of the input space) on dense zones, but their behaviour on sparse areas is unpredictable and depends on the learning technique. Also, these less dense zones are those more likely to

[1] Valencian Research Institute for Artificial Intelligence (vrAIn), Universitat Politècnica de València, Spain, email: {rafabbo,cferri,fmartinez,mramirez}@dsic.upv.es, jorallo@upv.es

[2] JRC, European Commission, email: fernando.martinez-plumed@ec.europa.eu

[3] Freely adapted from Jane Austen's "Pride and Prejudice".

[4] The non-optimality phenomenon is straightforward and less controversial than the particular assumptions (block-uniformity) and (stronger) results on expectation of the no-free-lunch theorems [35].

[5] https://cran.r-project.org/web/views/MachineLearning.html

[6] https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

[7] https://en.wikipedia.org/wiki/Outline_of_machine_learning

contain vulnerabilities [9, 10]. Another reason to incorporate sparse data lies in its potential pervasive effect on classifiers: it is usually the cause of negative effects on predictive accuracy [2, 23].

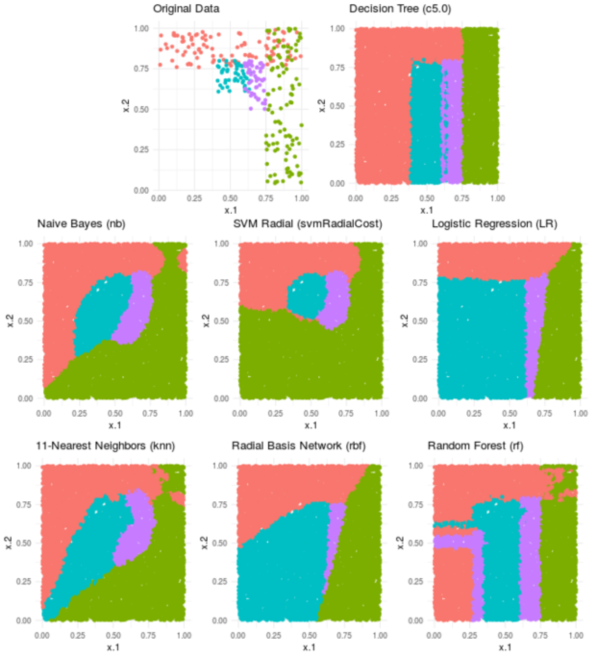

Figure 1: Different learning algorithms (classifiers) disagree on the elements near the boundaries, but more critically when generalising on sparse zones. The training data is shown on the top left.

We build our taxonomy using a combination of examples from dense and sparse areas, and using the *Cohen's kappa* coefficient (κ) [22] as the divergence metric for classification on a wide range of classification techniques (65) and problems (42 binary and 33 multiclass). The kappa statistic is a general-purpose agreement measure, and discounts agreements by chance. It is used not only to evaluate a single classifier, but also to evaluate classifiers amongst themselves. It is robust to data imbalance, especially when aggregating for multiple datasets [22].

We build a dissimilarity matrix by aggregating the results on the multiple datasets, we then cluster them into families using an agglomerative approach. In order to show the appropriateness of the proposed taxonomy, we conduct a series of experiments where we use different strategies to search for the best model over a new classification problem from a set of machine learning techniques. We also consider strategies based on the reputation of the learning technique. The results show that a search strategy based on exploring the diversity of behaviour based on the families we introduced is better than using some other folk family taxonomies, and much better than relying on the reputation of the techniques.

The contributions of the paper are the following:

- We analyse the divergence of machine learning techniques based on their experimental behaviour, using a combination of sparse and dense areas.
- Based on this study, we propose the first hierarchical family taxonomy that groups machine learning techniques according to their actual behaviour, instead of their formulation or name.
- We validate the proposed taxonomy over some folk taxonomies on a scenario where we iteratively select algorithms based on past performance (reputation), family diversity or both.

The rest of the paper is structured as follows. Section 2 reviews related work and further motivation. In Section 3 we detail the methodology we have used to calculate the similarity between techniques and the agglomerative clustering according to behaviour. The actual derivation of the families is shown in Section 4. In Section 5 we evaluate our proposed taxonomy in a scenario where we compare different strategies for model selection on unseen classification problems. Finally, the conclusions close the paper.

2 BACKGROUND

As mentioned in the previous section, the concept of family in machine learning is usually based on subjective notions, sustained on historical reasons, the learning strategy of the technique or their casual arrangement in textbooks or libraries. For instance, Domingos [5] grouped the Machine Learning techniques into five families (named tribes). These tribes include five types of models: connectionist, Bayesian, symbolist, evolutionary and analogizers. Machine Learning practitioners usually establish a number of model families that they consider appropriate. For instance, in a meta-learning scenario, Fernández-Delgado et al [12] evaluated 179 classifiers against a collection of 121 datasets to establish a comparison in behaviour, under the main goal of finding out which models are likely to perform better given a new dataset. The techniques, however, were grouped into 17 model families based on traditional criteria rather than behaviour. Martínez-Plumed et al. [24, 25, 26] studied the application of Item Response Theory (IRT) to supervised classification tasks. In this case, they employed 128 classifiers, explicitly grouped into 15 model families, also identified subjectively.

Other research directions have used behaviour to group or analysed machine learning techniques with some partial or limited notions of family. For instance, Inza et al [17] employed 14 well-known classifiers on 11 medical datasets to measure the differences in behaviour of the different classifiers. The behaviour is captured by means of a Bayesian Network learned on the output of the classifiers. Thus, the (dis)similarity of the models is given by the joint probability distribution of the class label predictions of the different classifiers. The authors grouped their classifiers into 5 predefined model families, to assert the conditional independence between the model families. For instance, Schrimpf et al. [29] measure how similar some artificial neural network architectures are to the brain's mechanism for object recognition. In particular, the so-called 'Brain Score', a composite of multiple neural and behavioural benchmarks (each one consisting on a metric applied to a specific set of experimental data) is used for scoring how brain-like ANNs are. Fernández et. al [11] analysed Genetics Based Machine Learning (GBML) algorithms for rule induction and grouped them into a taxonomy based on the chromosome representation. The performance of the algorithms and families are evaluated on both standard classification problems and classification with imbalanced data sets using accuracy and kappa measures. The authors first compared the algorithms within a family to select the best one as the family representatives, which were then used for comparing the performance among families.

The notion of family has also been used in the context of meta-learning. The goal of meta-learning, often defined as learning to learn [3, 32], consists in automatically recommending a learning technique (or a list of learning techniques) that are supposed to perform well for a new given dataset. To achieve so, a meta-learner needs some prior knowledge to learn from, i.e., it extracts the relationships between the intrinsic properties of previously known datasets and the results of evaluating them with different models. This information is used to build a meta-model to predict which model (or models) can offer good performance for the new given dataset. Some work in the context of meta-learning, use or define families. For instance, Smith et al. [30] proposed an empirical definition of instance hardness based on the classification behaviour of a set of learning algorithms. In order to calculate instance hardness, the authors first selected 20 ML algorithms. Then they are grouped into 9 families by applying a hierarchical clustering algorithm using the probability that they make different predictions (*Classifiers Output Difference (COD) measure*) as the distance between algorithms. The COD scores are calculated using test data that follow the *same* distribution observed in the training data. In this regard, family construction is guided by the primary goal of using them for instance hardness estimation. For this reason, the obtained families are only based on the partial behaviour they exhibit on the data space covered by the training data, and are not meant to be extrapolated beyond this use. Duin et al. [6] compared the differences between traditional models built in a feature space and models built on a dissimilarity (distance) representation in that each example is described as the vector of distances to the training examples. The authors applied 19 algorithms (13 based on feature space and 6 based on dissimilarity space) on 301 datasets and performed a hierarchical clustering on the classification errors obtained by 10 fold-cross-validation. The groups are used to analyse how models built on different space representation differ on classification performance. Again, these families are not meant to extrapolate beyond the particular study.

AutoML [31, 13, 20] is an area closely related to meta-learning that tries to provide methods and processes to make machine learning usable for non-experts and to improve efficiency and applicability of machine learning. Its ultimate goal is to automate the whole ML pipeline: preprocessing, model selection/optimisation, interpretation of the results, etc. In the case of model selection, methods based on meta-learning are usually employed. These methods try to optimise the model in the available algorithmic space. Regarding parameter optimisation, the strategies to find optimal configurations include grid search, random search or Bayesian search, with some refinements to alleviate the computational effort, including a proper initialisation of the parameter space, to establish a budget to limit the search time, or to avoid slow model-parameter combinations. These optimisations are restricted to one kind of algorithm at a time, but the notion of family reappears as the pipeline can do parameter optimisation on a diversity of algorithms or architectures.

In both meta-learning and AutoML, when the existing knowledge base is not directly applicable to determine a promising method based on the features of the analysed dataset, the search for the best algorithm in the available space of machine learning falls back again on the folk notion of model families. As we have previously explained, this categorisation is based mainly on the underlying constructs that were used to develop the learning technique, and not really on behavioural diversity. A clever strategy for finding strong models should consider the intrinsic behaviour of the learning techniques, and explore a diversity of behaviours, with special emphasis on trying to find the right technique for those problems that are unusual or special. This behaviour-grounded diversity is ignored by the most of the methods employed in meta-learning and AutoML search strategies.

Similarly, when practitioners do not use meta-learning or AutoML, they usually end up looking for a diverse set of techniques, following the folk taxonomies. This is also the case when building ensemble methods. In particular, stacking [34] builds a combination of models that have to be both good and diverse [21]. Note that diversity can be used when only one model is selected, by trying techniques one after another [7]. We will use this combination of goodness and diversity in one of the strategies that we will use to validate our taxonomy. Overall, the definition of a proper taxonomy of machine learning techniques could replace these traditional taxonomies and encourage the use of diversity in a more systematic way.

3 IDENTIFYING BEHAVIOUR-BASED ML FAMILIES

As stated previously, a popular and simple way to group ML techniques is based on the nitty-gritty of their formulation, such as whether they are based on a Bayesian approach, they use gradient descent or a discrete search space over rules. This approach is practical for machine libraries and textbooks, as the components and implementations cluster around these formulations, but may lead to similar behaviour. Actually, it is usually the case that one classifier of a family with some hyperparameters (or under some assumptions) is shown to be equivalent to a technique of a different family. Some theoretical connections are expressed in terms of bounds, but theoretically-similar methods may still differ dramatically depending on the dataset, and which area of the space we pay attention too.

Thus, a more appropriate way of analysing and comparing machine learning techniques is to studying their actual behaviour, i.e. whether they agree or disagree when classifying instances. However, in the literature, the behaviour of a model is usually checked against a test set that is supposed to be representative of the problem. In other words, the test set somewhat follows the distribution of the original training set. This is a valid setting when we want to check how well a model performs for a certain task, where the new incoming objects to identify should ideally be similar to those observed in the training set. However, if we want to study the actual behaviour of a set of models, we need to consider test data covering all the feature space, including sparse regions where we can find significant differences between models (see Figure 1). These sparse regions include examples which are outliers, thus presenting an unrealistic combination of features.

Accordingly, we propose to identify the families of learning algorithms (for classification) by focusing on their class agreement, i.e., the extent to which classifiers assign the same class to the same example. With this in mind, we use the *Cohen's kappa* coefficient (κ) [22]. Unlike other evaluation measures, the kappa coefficient is used not only to evaluate a single classifier, but also to evaluate classifiers amongst themselves (corrected by chance). Actually, with kappa we can also take into account how different classifiers disagree on boundaries caused by extrapolation noise.

Regarding the behaviour of a learning model, if we aim to consider how a model extrapolates its behaviour in those more sparse regions of the feature space (e.g., where there is not enough training data), we need further methods to generate the test data. The procedure for data generation is as follows: in the case of numerical attributes, we assume that their values are distributed according to a normal distribution (which we infer from the training set), so each artificial value is generated following such distribution. Note that the simulated numerical values generated are not bounded by a maximum nor a minimum when we generate a new value following a normal distribution. Consequently, outliers may appear beyond the limits observed in the training set for a specific feature. Analogously, for simulating the values for the categorical attributes we infer the probability of each value from the training data and then we follow a categorical distribution. Note that, in this way rare combinations of attribute values (not seen in the training set) still take place, as we ignore the joint distribution. This is convenient for our purposes because we want to check the behaviour of the model for those regions where there is no training data, and this does the job systematically.

The simulated data can be used to test and compare a large set of learning algorithms (following pairwise sequence comparisons) which, according to their results, can then be grouped applying a hierarchical clustering over their kappa values. More concretely, the process for (automatically) generating a behavioural taxonomy of ML techniques consists of the following steps: (1) we gather a collection of representative datasets for which we learn a wide variety of learning models. (2) For each dataset we generate a simulated artificial test set which is to be labelled by the different learned models. (3) We can then pairwise compare the behaviour of the different models by computing the kappa score between the outputs of the different models. We repeat this procedure for a large collection of datasets, and then compute the average kappa score obtained by each technique along the whole collection of datasets. (4) Finally, with the averaged kappa outcomes, we perform a distance-based clustering procedure, e.g., a hierarchical clustering [18, 1]. Hierarchical clustering techniques are intuitive and flexible, in the sense that they only need a distance metric for the analysed data and the result is graphically represented as a dendrogram, so we can appreciate how close some families are and choose the number of families in a more insightful way.Unlike other non-distance based clustering algorithms, hierarchical clustering does not require prior knowledge about the groups so we can select any number of clusters we find appropriate by interpreting the resulting dendrogram.

In the following section we report the experiments performed for obtaining the taxonomy of ML techniques using the above procedure.

4 EMPIRICAL DERIVATION OF THE TAXONOMY

In this section we describe the experiments performed to obtain a taxonomy of classification algorithms according to the behaviour of the models they generate. This is based on the agreements and disagreements in their output, which is measured by the Cohen's kappa coefficient, as explained in the previous section. We start introducing the set of representative datasets and learning models we have employed. Then, we describe the experimental setting and we show and analyse the obtained results[8].

4.1 Data and Classifiers

To conduct our experiments, we started with an initial collection of representative datasets from the OpenML repository [33]. Concretely, we used those datasets from the *study 14*, the most comprehensive, collaborative and reproducible analysis from OpenML, containing a representative set of 100 public datasets[9]).

For each dataset, we performed a data cleansing procedure to avoid incomplete and inconsistent data which would affect the learning algorithms, following the usual procedure in similar works [12]. Specifically, we (1) remove those attributes having a large amount of missing values (e.g., higher than 25%), (2) remove 0-variance attributes, (3) remove instances (rows) with missing values, and (4) remove duplicated instances having different labels (e.g., noisy instances). After this process, we discarded those datasets for which we lost more than 30% of attributes or examples. With this, we end up having 75 datasets (Table 1). The numerical features are also standardised to have zero mean and standard deviation one, as in [12].

We use a wide variety of learning algorithms (classifiers), all of them implemented in R [27]. We try to cover the large collection of techniques available in `CARET`[10] and `RWeka`[11] R packages for classification tasks. Although we keep the default (hyper-)parameters to a large extent, we modified the tuning parameters for some classifiers in such a way that we obtained a heterogeneous set of 65 different classifiers (see Table 2 for a description).

Table 1: Datasets remaining after the cleansing procedure. There are 75 datasets in total: 42 binary datasets and 33 multiclass datasets.

Dataset	**#class**	**#feats**	**#ex**	Dataset	**#class**	**#feats**	**#ex**
abalone	28	8	4177	mfeat-zernike	10	47	2000
ada_agnostic	2	48	4562	monks1	2	6	556
analcatdata_authorship	4	70	841	monks2	2	6	601
artificial-characters	10	7	10218	monks3	2	6	554
badges2	2	10	294	mushroom	2	22	8124
balance-scale	3	4	625	musk1	2	167	476
banknote-authentication	2	4	1372	one-hundred-plants-margin	100	64	1600
blood-transfusion-service-center	2	4	748	one-hundred-plants-shape	100	64	1600
breast-cancer	2	9	286	one-hundred-plants-texture	100	64	1599
breast-w	2	9	699	optdigits	10	64	5620
bupa	2	6	345	ozone	2	72	2536
car	4	6	1728	ozone-level-8hr	2	72	2534
cardiotocography	10	35	2126	pc1	2	21	1109
chess-KRVKP	2	36	3196	pc3	2	37	1563
climate-model-simulation-crashes	2	20	540	pc4	2	37	1458
cmc	3	9	1473	pendigits	10	16	10992
credit-a	2	15	690	PhishingWebsites	2	30	11055
credit-g	2	20	1000	phoneme	2	5	5404
diabetes	2	8	768	qsar-biodeg	2	41	1055
eucalyptus	5	19	736	satimage	6	36	6430
first-order-theorem-proving	6	51	6118	scene	2	299	2407
GesturePhaseSegmentationProcessed	5	32	9873	segment	7	19	2310
heart-statlog	2	13	270	semeion	10	256	1593
hepatitis	2	19	155	sonar	2	60	208
hill-valley	2	100	1212	spambase	2	57	4601
ilpd	2	10	583	spect_full	2	22	267
ionosphere	2	34	351	splice	3	60	3190
iris	3	4	150	steel-plates-fault	2	33	1941
JapaneseVowels	9	14	9961	synthetic_control	6	60	600
jm1	2	21	10885	texture	11	40	5500
kc1	2	21	2109	tic-tac-toe	2	9	958
kc2	2	21	522	vehicle	4	18	846
liver-disorders	2	6	345	vowel	11	12	990
lung-cancer	3	56	32	wall-robot-navigation	4	24	5456
mfeat-factors	10	216	2000	waveform-5000	3	40	5000
mfeat-fourier	10	76	2000	wdbc	2	30	569
mfeat-karhunen	10	64	2000	wilt	2	5	4839
mfeat-morphological	10	6	2000				

Table 2: List of the 65 classifiers employed in the experiments, along with the parameters we used. Parameters are left to their default values, unless specified otherwise.

Technique (parameters)	**Description**
AVNNet (decay = 1e02) AVNNet (decay = 1e04)	Averaged NNet
BagFDA (prune = 4) BagFDA (prune = 8)	Bagged Flex. Disc. Analysis
Bagging	Bagging
C5.0 C5.0 (winnow)	C5.0 tree–based model
CI_RF (mtry = 16) CI_RF (mtry = 64)	Cond. Inf. Random Forest
CI_T (mincriterion = 0.01) CI_T (mincriterion = 0.05)	Cond. Inf. Tree
MARS	Mult. Adap. Reg. Splines
FDA (nprune = 17) FDA (nprune = 9)	Flex. Disc. Analysis
GBM (id = 2, ntrees = 100) GBM (id = 2, ntrees = 50)	Stoch. Grad. Boosting Machine
HDRDA	HighDim. Reg. Disc. Analysis
Ibk (k = 3) Ibk (k = 5)	Instance–based learning algorithm
J48 J48 (unpruned)	C4.5 decision tree algorithm
JRip JRip (unpruned)	Decision rules
3–NN 5–NN	k-nearest neighbors classifier
LMT	Log. Model Trees
LR	Log. Regression
LVQ (K = 3) LVQ (K = 5)	Learning Vector Quant.
MLP (HL = 1, U = 3) MLP (HL = 1, U = 5) MLP (HL = 1, U = 7) MLP (HL = 2, U = 3)	MultiLayer Perceptron
MLP (HL = 1, U = 5, WD = 1e05) MLP (HL = 2, U = 5, WD = 1e05)	MultiLayer Perceptron
PMR	Penalized Multinomial Regression
NB NB (Lapace Smoothing = 3) NB (RWeka)	Naive Bayes
ParRF (mtry = 16) ParRF (mtry = 64)	Parallel Random Forest
PART	Rule Sets
PDA	Penalized Disc. Analysis
GPS	Greedy Prototype Selection
RBF RBF (dynamic decay adjustment)	Radial Basis Func. Network
RDA	Reg. Disc.Analysis
RF (mtry = 16) RF (mtry = 64)	Random Forest
RFRules (mtry = 16) RFRules (mtry = 64)	Random Forest Rule–Based
RPART	Rec. Part. And Reg. Trees
RegRF (mtry = 16) RegRF (mtry = 64)	Reg. Random Forest
SDA (lambda = 1)	Shrinkage Disc. Analysis
PLS (ncomp = 2) PLS (ncomp = 3) PLS (ncomp = 4)	Part. Least Squares
SVM (Linear, C = 1) SVM (Linear, C = 2) SVM (Poly, degree = 1) SVM (Poly, degree = 2) SVM (Poly, degree = 3) SVM (RBF, cost = 2ˆ11) SVM (RBF, cost = 2ˆ11)	Support Vector Machine

4.2 Hierarchical Clustering

We then perform pairwise comparisons between the outputs of the different trained models per dataset. We created a triangular matrix

[8] For the sake of reproducibility and replicability, all the data, code, complete experiments, plots and results can be found in https://github.com/rfabra/model_family_taxonomy

[9] https://www.openml.org/s/14

[10] http://topepo.github.io/caret/index.html

[11] https://cran.r-project.org/web/packages/RWeka/index.html

of size $N \times N$, where N is the number of learning techniques (in our experiments N=65). Each cell represents the average (1-$kappa$) (for all datasets) obtained by a pairwise comparison of the outputs of the models induced with the corresponding techniques. We use $1 - kappa$ since clustering methods are based on distance matrices. Then, we applied a hierarchical clustering algorithm [19] to group the different models, using the complete distance in order to find the maximum possible distance between points belonging to two different clusters (this linkage method tends to produce more compact clusters [8]).

4.3 Results

From the hierarchical cluster analysis using the proposed 75 datasets, we obtained the dendrogram shown in Figure 2 (left). As a clarification, in a dendrogram, each leaf corresponds to one observation (e.g., one learning algorithm). As we move up the tree, observations that are similar to each other are combined into branches, which are themselves fused at a higher height. The height of the fusion, provided on the horizontal axis, indicates the (dis)similarity between two observations. The higher the height of the fusion, the less similar the observations are.

Since 54% of the datasets (41 out of 75) are binary, we have also analysed the potential groups arising from those 2-class (Figure 2, centre) as well as those using only multiclass datasets (Figure 2, right). In order to identify subgroups (i.e., clusters), we may cut the dendrograms at a certain height. In this regard, we grouped the models into 18 clusters (i.e., algorithm families). This cut value represents a reasonable compromise between having a significant number of families of techniques and getting well defined groups (techniques in the same group are much more similar among them than to those in other groups). We also check the cohesion and separation of the clusters performing the *Silhouette* [28] method (a procedure for the interpretation and validation of consistency within clusters of data). It is a very common measure that performs well in cases where there is no centroid or boundaries between clusters, as in this case. The *Silhouette* method determined the optimal number of clusters to be equal to 18 (see Figure 3).

The dendrograms of Figure 2 are useful to show clearly how much resemblance there is in the behaviour of the considered models. From Figure 2 (left) we can see that the cluster A_1 corresponds to the neural networks family. All the *Multi-Layer Perceptron* algorithms, under different configurations, along with the model *Average Neural Network* are grouped together. We can also appreciate that *Logistic Model Trees* is included within this group, but at a higher height. In fact, this model is quite close to the following group, A_2, which correspond to linear models. This latter cluster also includes *Logistic Regression*-based methods, *SVM* with linear kernels and the *Penalized Discriminant Analysis* techniques. The group A_3 contains algorithms such as the *Flexible Discriminant Analysis* with different levels of pruning, along with the *MARS* technique (*Multivariate Adaptative Regression Splines*). The cluster A_4 corresponds to *Partial Least Squares*-related methods (varying the number of components), and the group A_5 is formed by two methods based on *Regularized Discriminant Analysis*. Instance-based methods (*k-NN*, *Ibk* and *Greedy Prototype Selection*) also fall under the same group, A_6. Close to these, we find the *Learning Vector Quatization*-related methods with varying K parameters (A_7). We find the *SVM* group (A_8) with non-linear kernels: polynomial kernels with degree 2 and 3, and radial basis function kernels. Next we find several groups containing different types of ensembles and trees. We can see that *Stochastic Gradient Boosting Machines*, *Conditional Inference Random Forests* and *Bagged Flexible Discriminant Analysis* fall under the same family, A_9. The *CART* model is grouped with the *Bagging* model in A_{10}. Although this might look counterintuitive, we can see, however, that they are combined into branches at a high height, and they are quite close to the previous ensembles, as well as the family A_{11}, composed by pruned and unpruned *JRip* models. *Conditional Inference Trees* also have their own family, A_{12}. The cluster A_{13} corresponds with decision tree-related models, containing different configurations of both *C5.0* and *J48* algorithms. The *PART* model has its own group. However, it is about to fall under the previous group, so we could consider that whole branch as the same family. The group A_{15} corresponds with different types of *Random Forest*. We have the cluster A_{16} composed of two configurations of *Random Forest Decision Based Rules*. Despite having their own family, we can see that these models are closely related to tree and tree ensemble models. The two last families correspond to *Naive Bayes* models (along with *Shrinkage Discriminant Analysis*) and *Radial Basis Function Network* models.

If we pay attention to binary datasets (Figure 2, centre), the dendrogram depicts some changes.The *Shrinkage Discriminant Analysis* (B_2) is now separated from *Naive Bayes* (its potential stereotyped family). The *Greedy Prototype Selection* algorithm also has its own group, which is close to the group B_4 (*Nearest Neighbours*). *SVM* with Polynomial kernel (degrees 2 and 3) fall in the *Neural Networks* cluster, B_7. *SVM* with radial basis function Kernels are grouped together (B_8), close to the *Neural Networks* family. Regularised *Discriminant Analysis* models are now in the same group as *Partial Least Squares* (B_9). Linear models, B_{10}, are kept as well. *Bagging*, *CART* and both versions of *J48* and *C5.0* are grouped together in B_{11}. This group is quite close to the *PART* model, which keeps its own group. Close to these two previous clusters we find a group of *Conditional Inference Trees*, B_{13}. *Flexible Discriminant Analysis* and *Multivariate Adaptive Regression Splines* are still grouped together, but *Logistic Model Trees* now fall also within this group, B_{14}. The rest of groups remain largely unchanged.

The dendrogram for the multiclass datasets (Figure 2, right) also presents small differences, although the groups clustered are more or less the same: *Naive Bayes* algorithms (C_1), *Radial Basis Function* networks (C_2), *JRip* models (C_{14}), *Random Forest Rule-based* techniques (C_{18}) and *Conditional Inference Trees* (C_{13}). Unlike the binary case, *Partial Least Squares* composes its own group (C_3), as well as *Regularized Discriminant Analysis*-related algorithms (C_5). There are also some results that may look counterintuitive. For instance, *SVM* with linear kernels fall in the same family (C_4) as *SVM* with radial basis function kernels. *SVM* with polynomial kernel (degrees 2 and 3) are together with *Nearest Neighbours* and *Greedy Prototype Selection* (C_7). One of the *Neural Network* models (*Model Averaged Neural Network*) have their own group (C_9), since the distance to the other neural networks is slightly higher. *Logistic (and multinomial) Regression* are now in their own group, C_{10}, along with *Penalized Discriminant Analysis*. *Logistic Model Trees* are very close to this group. There is another big group, C_{12} composed by *CART* and tree ensembles (all types of *Random Forest* and *Stochastic Gradient Machines*). *Flexible Discriminant Analysis* and *Multivariate Adaptative Regression Splines* are kept in the same family (C_{15}). Contrary to the previous cases, *Bagged Flexible Discriminant Analysis* have their own group, C_{16}. The model *PART* is now together with decision trees (*C5.0*, *J48* and their variants, in C_{17}).

As we can see, there are few differences between the results for binary datasets and for all datasets. However, there are relevant differ-

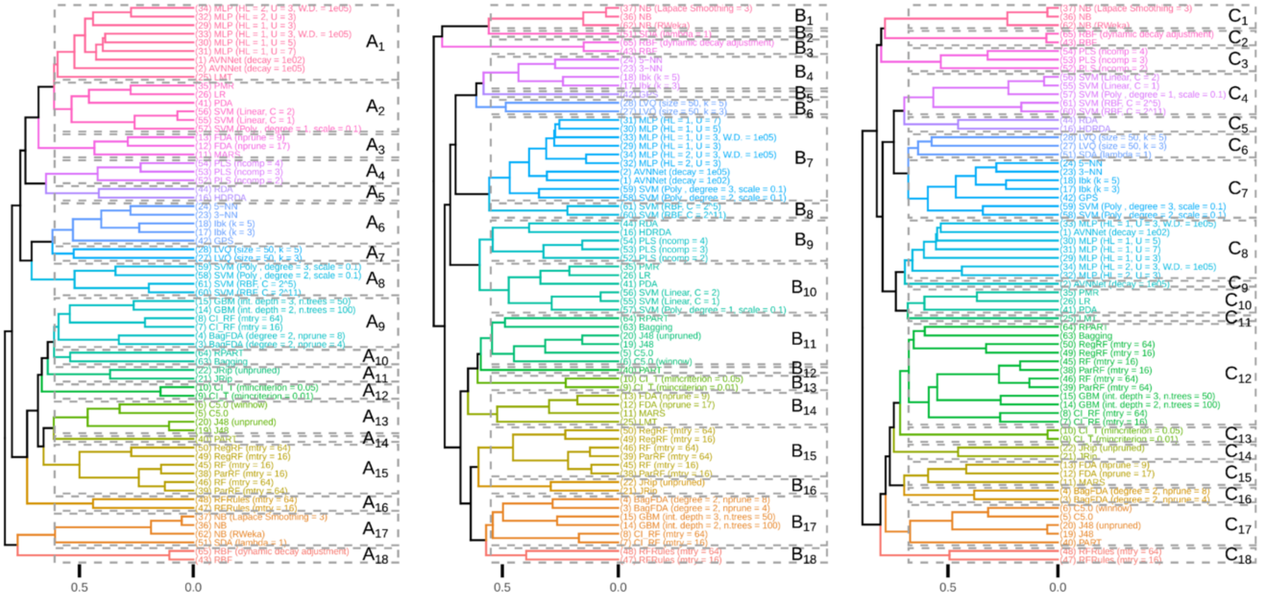

Figure 2: Model family dendrogram representing the hierarchical clustering of machine learning models from Table 2. (Left) Dendrogram obtained using the 75 datasets (binary and multiclass). (Centre) Dendrogram obtained using only binary datasets. (Right) Dendrogram obtained using only multiclass datasets.

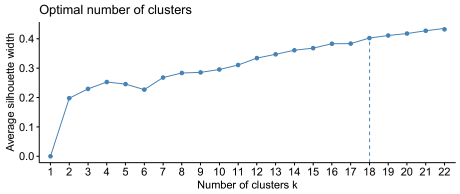

Figure 3: Silhouette criterion reached in 18 groups when clustering AI algorithms according to their behaviour.

ences between these results and the results for the multiclass datasets. This might happen due to the way in which some models, defined for binary problems (e.g., SVM models), generalise to the multiclass case. We must also take into account that, due to our test generation method, we are probably covering many regions of the feature space that are not covered with the training set. This may also explain why some models that are supposed to behave differently (like SVM with different kernels) present similar behaviour, which can be because the models extrapolate very similarly to the sparse regions.

5 VALIDATION USING TECHNIQUE SELECTION

In this section we validate our taxonomy of machine learning families by performing a number of experiments for technique selection. More specifically, we address the problem of choosing the right machine learning algorithm among a vast amount of possibilities when addressing a new learning problem (e.g., a new, previously unseen, dataset), and without considering (or knowing) any intrinsic characteristic of the task at hand. As the best algorithm depends on the data, the best choice of the learning algorithm remains unclear unless we test out all our algorithms directly on the data. However, we can improve this search of the best technique in different ways: (1) according to the reputation they have for other problems, or (2) based on the behavioural family diversity.

In this regard, we have implemented five different algorithm selection approaches. We applied them to a subset of 50 new datasets[8] from the OpenML repository [33] (see Table 3 for a description). The available portfolio of candidate algorithms to be used are those 65 different classifiers we describe in section 4.2. These approaches are:

Table 3: Description of the 50 datasets used in the validation phase.

Dataset	**#class**	**#feats**	**#ex**	Dataset	**#class**	**#feats**	**#ex**
arsenic-female-bladder	2	4	559	no2	2	7	500
bank32nh	2	32	8192	parkinsons	2	22	195
blogger	2	5	100	pollen	2	5	3848
calendarDOW	5	32	399	prnn_crabs	2	7	200
car-evaluation	4	21	1728	puma32H	2	32	8192
CastMetal1	2	37	327	puma8NH	2	8	8192
churn	2	20	5000	ringnorm	2	20	7400
cpu_act	2	21	8192	sa-heart	2	9	462
delta_ailerons	2	5	7129	seeds	3	7	210
delta_elevators	2	6	9517	seismic-bumps	3	7	210
dna	3	180	3186	servo	2	4	167
eye_movements	3	26	10936	sleuth_case2002	2	6	147
fri_c4_100_10	2	10	100	Smartphone-Based_Recognition	6	66	180
fruitfly	2	4	125	socmob	2	5	1156
glass	6	9	214	solar-flare_1	5	12	315
grub-damage	2	8	155	space_ga	2	6	3107
hayes-roth	2	4	132	threeOf9	2	9	512
jungle_chess	3	46	4704	vertebra-column	3	6	310
kin8nm	2	8	8192	veteran	2	7	137
led24	10	24	3200	visualizing_galaxy	2	4	323
lymph	4	18	148	visualizing_soil	2	4	8641
machine_cpu	2	6	209	wind	2	14	6574
MindCave2	2	39	125	wine-quality-red	6	11	1599
new-thyroid	3	5	215	wine	3	13	178
newton_hema	2	3	140	yeast	10	8	1484

- **Random selection** (*Random*): The selection of the algorithms to evaluate against the different datasets is made at random. The same shuffle of classifiers is applied over all the datasets. We perform 100 repetitions and average the results.
- **Reputational selection** (*Reputation*). We apply all the models following the order given by the ranking, from best to worst. First, we use an extra set of 10 datasets (different from the above 50) from which obtain an algorithm ranking order. This ranking is based on the evaluation accuracy for the set of 65 machine learning algorithms, using a 70-30% splitting criterion for each dataset and averaging the results.
- **Behavioural Family Diversity** (*BF-Diversity*): Using the previous order of models, we add diversity in the selection procedure by grouping the algorithms by their family. The algorithms are thus selected by their ranking as well as their family: if the model to be used (according to the ranking provided) belongs to a family

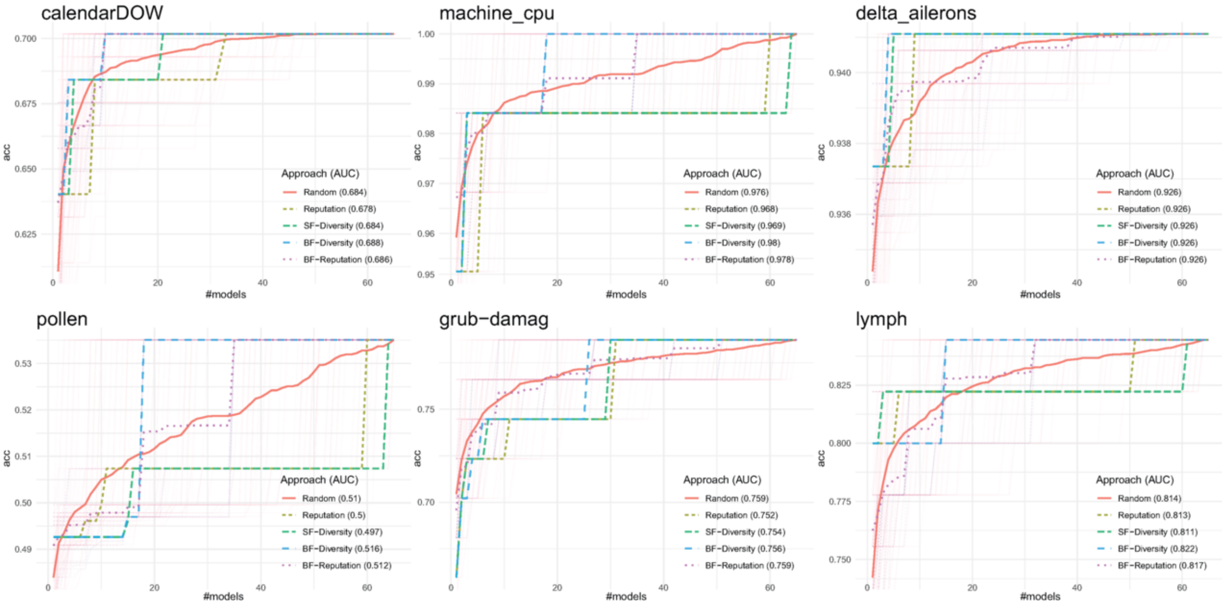

Figure 4: Illustrative selection of 6 datasets (from those 50 used in the validation procedure) showing the application of our algorithm selection approaches. Each plot shows the accuracy obtained against the number of models tested. Semi-transparent lines represent 100 repetitions for those approaches with random components. AUC values (scaled between 0 and 1) for each approach are provided in the legend.

already used (in previous selections), we jump to the next model in the ranking. If all the families have been used, we start over again using the ranking but this time selecting the next best algorithm. This is repeated until all the algorithms have been tried and evaluated.

- **Behavioural Family Reputation** (*BF-Reputation*): This is similar to the previous approach, but we only use the order provided by the ranking to sort our families. We then select the learning algorithms at random: one by family and starting over again when all families have been tried, without replacements. We average the results over 100 repetitions.
- **Stereotyped Family Diversity** (*SF-Diversity*). This is similar to the *BF-Diversity* approach, but instead of using our taxonomy of families, we use the set of stereotyped families defined in [12], the most comprehensive empirical evaluation of machine learning algorithms up until now. The classifiers used in this study are grouped by similarity in 17 families (discriminant analysis, Bayesian, neural networks, support vector machines, decision trees, rule-based classifiers, boosting, bagging, stacking, random forests and other ensembles, generalised linear models, nearest neighbours, partial least squares and principal component regression, logistic and multinomial regression, multiple adaptive regression splines and other methods)[8].

The idea is to follow the usual trial and error procedure checking the accuracy of an ordered selection of algorithms. In this regard, Figure 4 shows the performance for each of the above approaches using 6 illustrative datasets (picked out from the 50 datasets)[12]. While the x-axis shows the number of algorithms used (1 to 65), the y-axis shows the maximum accuracy among the set of techniques already used in each point in the x-axis. In order to analyse the performance between the different approaches, we consider the area under the curve of the maximum reached accuracy. The results can be found in Table 4, where we perform a pairwise comparison (wins, ties and losses) between all the selection approaches implemented. We can see that reputational-based approaches (e.g., the reputation/performance they have for other problems) are worse than selecting techniques based on family diversity. We also show that, when diversity is required, our taxonomy of families based on (dis)agreement in behaviour is also better than those groupings of algorithms made according to stereotypes (in terms of implementation). The results are split according to the size of the 50 datasets addressed: large/small dataesets are those containing more/less examples than the average over all the datasets. We see clear differences among these two groups, where the larger is the size of the datasets, the more pronounced are the benefits of using our approach based on family diversity.

Table 4: Pairwise comparison (row **W**ins - **T**ies - **L**oses against column) between selection approaches based on family diversity (*BF-Diversity* and *BF-Reputation*), and baselines using the similarity-based grouping in [12] (*SF-Diversity*), as well as *Reputation* and *Random* selection procedures. Results split by the size of the datasets used. Wins in bold when statistically significant using the two-sided binomial test at 95% confidence level (with ties counted as half).

	Large datasets (W-T-L)			
	BF-Reputation	**SF-Diversity**	**Reputation**	**Random**
BF-Diversity	**16** - 0 - 1	9 - 5 - 3	**13** - 3 - 1	**16** - 0 - 1
BF-Reputation		8 - 2 - 7	11 - 0 - 6	**15** - 0 - 2
SF-Diversity			9 - 3 - 5	**16** - 0 - 1
Reputation				**16** - 0 - 1
	Small datasets (W-T-L)			
	BF-Reputation	**SF-Diversity**	**Reputation**	**Random**
BF-Diversity	**24** - 1 - 8	19 - 6 - 8	**24** - 3 - 6	**27** - 0 - 6
BF-Reputation		14 - 2 - 17	18 - 2 - 13	**26** - 0 - 7
SF-Diversity			18 - 9 - 6	**27** - 0 - 6
Reputation				**23** - 0 - 10

6 CONCLUSIONS

In this paper we proposed a method to identify ML families. The methodology was based on comparing the behaviour of 65 different

[12] The rest of experiments and plots can be found in https://github.com/rfabra/model_family_taxonomy

learning models (including hyperparameter variations), performing a pairwise comparison (based on Kappa) and averaging the results obtained for 75 datasets. We applied a hierarchical clustering so that the models that presented similar behaviour fell in the same cluster, which we considered a model family. This method allowed to objectively quantify how different two models (or model families) are.

To compare the models between them, we generated artificial datasets designed to cover the regions of the feature space with sparse (or no) training data. With this we consider how the models extrapolate their boundaries for regions unseen in the training phase. This strategy revealed some interesting observations. For instance, the dendrogram obtained for the multiclass case (Figure 2, right) shows that some of the SVM with different kernels (linear and RBF kernels) are grouped into the same family, which may look counterintuitive. The results also show that the final families differ between multiclass and binary problems. This may happen because the more classes a problem has, the greater the differences will be in their extrapolation of the boundaries to the areas with few or no training data.

In this work we generated different models from a wide range of learning techniques and hyperparameters. However, it is also possible to apply our method for a single learning technique under a wide range of parameter variations (for instance, different architectures or hyperparameters of neural networks) to establish the similarity in behaviour of the different combinations. The same procedure could be applied when varying different preprocessing options, for instance, we could analyse the models and see the differences in behaviour when applying different techniques to deal with missing values. Practitioners can also employ our families in different areas such as in meta-learning settings to establish similarities between the different models that a meta-learner may recommend, or in adversarial learning scenarios in order to use common defense techniques in models with similar behaviour [10, 9]. Finally, as future work we would like to apply a similar approach to regression techniques.

Overall, this paper vindicates the use of algorithm families as a useful construct in the arrangement of experiments, the analysis of classical and new algorithms, its use in metalerning, AutoML and model combination, and other areas of machine learning. To encourage this use we have presented a practical taxonomy for classification. More importantly, the methodology can be used for deriving or refining new taxonomies as new techniques and tasks are considered.

ACKNOWLEDGEMENTS

This research was supported by the EU (FEDER), Spanish MINECO (RTI2018-094403-B-C32), Generalitat Valenciana (PROMETEO/2019/098) and the AirForce Office of Scientific Research (FA9550-17-1-0287). JHO is also funded by FLI (RFP2-152). FMP is supported by INCIBE, the European Commission (CT-EX2018D335821-101) and UPV (PAID-06-18).

REFERENCES

[1] P. Berkhin, 'A survey of clustering data mining techniques', in *Grouping multidimensional data*, 25–71, Springer, (2006).

[2] J. Bissmark and O. Wärnling, *The Sparse Data Problem Within Classification Algorithms: The Effect of Sparse Data on the Naïve Bayes Algorithm*, Ph.D. dissertation, KTH, 2017.

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

[4] R. Caruana and A. Niculescu-Mizil, 'An empirical comparison of supervised learning algorithms', in *ICML*, pp. 161–168. ACM, (2006).

[5] P. Domingos, *The master algorithm: How the quest for the ultimate learning machine will remake our world*, Basic Books, 2015.

[6] R. Duin, M. Loog, E. Pekalska, and D. Tax, 'Feature-based dissimilarity space classification', in *Recognizing Patterns in Signals, Speech, Images and Videos*, 46–55, Springer, (2010).

[7] S. Džeroski and B. Ženko, 'Is combining classifiers with stacking better than selecting the best one?', *Mach. Learning*, **54**(3), 255–273, (2004).

[8] B. Everitt, S. Landau, and M. Leese, *Cluster Analysis*, Wiley, 2009.

[9] R. Fabra-Boluda, C. Ferri, J. Hernández-Orallo, F. Martínez-Plumed, and M. Ramírez-Quintana, 'Modelling machine learning models', in *PT-AI*, pp. 175–186. Springer, (2017).

[10] R. Fabra-Boluda, C. Ferri, J. Hernández-Orallo, F. Martínez-Plumed, and M. Ramírez-Quintana, 'Identifying the machine learning family from black-box models', in *CAEPIA*, pp. 55–65. Springer, (2018).

[11] A. Fernández, S. García, J. Luengo, E. Bernadó-Mansilla, and F. Herrera, 'Genetics-based machine learning for rule induction: state of the art, taxonomy, and comparative study', *IEEE Trans. on Evolutionary Computation*, **14**(6), 913–941, (2010).

[12] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, 'Do we need hundreds of classifiers to solve real world classification problems', *J. Mach. Learn. Res*, **15**(1), 3133–3181, (2014).

[13] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, 'Efficient and robust automated machine learning', in *NIPS*, 2962–2970, (2015).

[14] P. Flach, *Machine learning: the art and science of algorithms that make sense of data*, Cambridge University Press, 2012.

[15] D. Gunning, 'Explainable artificial intelligence (xai)', *DARPA, nd Web*, **2**, (2017).

[16] J. Hernández Orallo, C. Ferri Ramírez, and M. Ramírez Quintana, *Introducción a la Minería de Datos*, Pearson Prentice Hall, 2004.

[17] I. Inza, P. Larrañaga, et al., 'Representing the behaviour of supervised classification learning algorithms by bayesian networks', *Patt. Rec. Letters*, **20**(11-13), 1201–1209, (1999).

[18] A. K Jain, N. Murty, and P. Flynn, 'Data clustering: a review', *ACM computing surveys*, **31**(3), 264–323, (1999).

[19] L. Kaufman and P. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*, volume 344, John Wiley & Sons, 2009.

[20] L. Kotthoff, C. Thornton, H. H Hoos, F. Hutter, and K. Leyton-Brown, 'Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka', *JMLR*, **18**(1), 826–830, (2017).

[21] L. Kuncheva, *Combining pattern classifiers: methods and algorithms*, John Wiley & Sons, 2014.

[22] R. Landis and G. Koch, 'An application of hierarchical kappa-type statistics in the assessment of majority agreement among multiple observers', *Biometrics*, 363–374, (1977).

[23] X. Li, C. Ling, and H. Wang, 'The convergence behavior of naive bayes on large sparse datasets', *ACM TKDD*, **11**(1), 10, (2016).

[24] F. Martínez-Plumed et al., 'IRT in AI: Analysing machine learning classifiers at the instance level', *Artificial Intelligence*, **271**, 18–42, (2019).

[25] F. Martinez-Plumed and J. Hernandez-Orallo, 'Dual indicators to analyse ai benchmarks: Difficulty, discrimination, ability and generality', *IEEE Transactions on Games*, (2018).

[26] F. Martınez-Plumed, R. Prudêncio, A. Martınez-Usó, and J. Hernández-Orallo, 'Making sense of item response theory in machine learning', in *ECAI*, volume 285, pp. 1140–1148, (2016).

[27] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2019.

[28] P. Rousseeuw, 'Silhouettes: a graphical aid to the interpretation and validation of cluster analysis', *J. Comput. Appl. Math.*, **20**, 53–65, (1987).

[29] M. Schrimpf et al., 'Brain-score: which artificial neural network for object recognition is most brain-like?', *BioRxiv*, 407007, (2018).

[30] Michael R Smith, Tony Martinez, and Christophe Giraud-Carrier, 'An instance level analysis of data complexity', *Machine learning*, **95**(2), 225–256, (2014).

[31] A. Truong et al., 'Towards automated ml: Evaluation and comparison of AutoML approaches and tools', *arXiv:1908.05557*, (2019).

[32] J. Vanschoren, 'Meta-learning: A survey', *arXiv:1810.03548*, (2018).

[33] J. Vanschoren, J. Van Rijn, B. Bischl, and L. Torgo, 'OpenML: networked science in machine learning', *ACM SIGKDD Explorations Newsletter*, **15**(2), 49–60, (2014).

[34] D. Wolpert, 'Stacked generalization', *Neural networks*, **5**(2), 241–259, (1992).

[35] D. Wolpert, 'What the no free lunch theorems really mean; how to improve search algorithms', in *Santa Fe Institute*, volume 4, (2012).