entropy

Article

Instance Selection for Classifier Performance Estimation in Meta Learning

Marcin Blachnik

Department of Applied Informatics, Silesian University of Technology, 44-100 Gliwice, Poland; marcin.blachnik@polsl.pl; Tel.: +48-32-603-4170

Received: 20 September 2017; Accepted: 23 October 2017; Published: 1 November 2017

Abstract: Building an accurate prediction model is challenging and requires appropriate model selection. This process is very time consuming but can be accelerated with meta-learning–automatic model recommendation by estimating the performances of given prediction models without training them. Meta-learning utilizes metadata extracted from the dataset to effectively estimate the accuracy of the model in question. To achieve that goal, metadata descriptors must be gathered efficiently and must be informative to allow the precise estimation of prediction accuracy. In this paper, a new type of metadata descriptors is analyzed. These descriptors are based on the compression level obtained from the instance selection methods at the data-preprocessing stage. To verify their suitability, two types of experiments on real-world datasets have been conducted. In the first one, 11 instance selection methods were examined in order to validate the *compression–accuracy* relation for three classifiers: *k*-nearest neighbors (*k*NN), support vector machine (SVM), and random forest. From this analysis, two methods are recommended (*instance-based learning* type 2 (*IB2*), and *edited nearest neighbor* (*ENN*)) which are then compared with the state-of-the-art metaset descriptors. The obtained results confirm that the two suggested compression-based meta-features help to predict accuracy of the base model much more accurately than the state-of-the-art solution.

Keywords: machine learning; classification; instance selection; meta-learning; accuracy estimation

1. Introduction

The data mining process consists of four stages, which are: (1) data collection; (2) data preprocessing (of all data transformation elements, for example feature normalization, instance or feature selection [1], discretization [2], and type conversion); (3) prediction model construction (e.g., *k*-nearest neighbors (*k*NN), decision trees, and kernel methods) [3]; and finally (4) data postprocessing (see Figure 1). In real applications, when an accurate model is needed the process requires a search for the best combination among the elements available in stages (2) and (3). This significantly increases the computational complexity (we have to find the combination of preprocessing elements which best supports given prediction model) thus causing a rise in the expenses of building the prediction system and a longer computational time.

There are several solutions to this problem (for example [4]), but one of the most robust solutions is meta-learning [5], which is designed to accelerate stage (3). It is based on representing the training set, which consists of *n* feature vectors by a single meta-date instance, and then uses it as an input to the meta-model which returns the estimate of the accuracy without training the actual (base) model [6]. The obtained values for different base models are ranked from the best to the worst and only the best model with the highest estimated accuracy is finally trained on the entire training set.

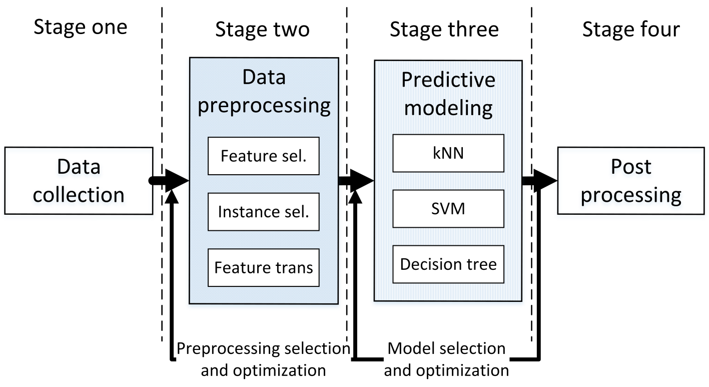


Figure 1. The data mining process with search loops, firstly to optimize predictive model type and its parameters, and secondly to optimize and select preprocessing operations. *k*NN: *k*-nearest neighbors; SVM: support vector machine.

The quality of the meta-learning system greatly depends on the quality of the meta-descriptors. In the beginning, these were simple statistics calculated per attribute or information theory measure (e.g., information gain) which can be obtained very quickly but usually are not very informative. Great progress was made with the introduction of landmark descriptors [7]. They are based on performances obtained by simple and "fast" algorithms like naive Bayes, decision trees, or one nearest neighbor (*1*NN). These descriptors are very informative and provide important insights into the nature of the data, but on the other hand this radically increases computational complexity. The research to identify good data descriptors is still open and an overview is provided in Section 3.

In this paper we also address this topic, but in contrast to the other researchers we postulate utilizing knowledge which can be extracted from the preprocessing stage, from the instance selection methods. These type of methods are commonly used for two purposes: data cleansing, and training set size reduction by eliminating redundant training samples. In other words, instance selection methods take as an input training set $\mathbf{T}$ and return a new dataset $\mathbf{P}$ such that $\mathbf{P} \subset \mathbf{T}$. In this paper we show that the relation between $\mathbf{T}$ and $\mathbf{P}$ is a very good landmark for the meta-learning system for example by considering the training set compression (see Equation (1)). Intuitively, when the training set $\mathbf{T}$ can be significantly compacted, the classification problem should not be difficult. High classification accuracy should be easy to achieve, and in the opposite case, with low compression, the problem is expected to be difficult and low accuracy can be expected. An inverse relation appears for cleansing methods where the increase in compression corresponds to a deterioration of the accuracy. The main thesis of this paper that compression of selected instance selection methods is a good meta-data descriptor for estimating the accuracy of selected machine learning methods. Thus, we reuse information extracted from the preprocessing stage that was obtained without any computational effort, and use it to construct new landmarks.

As mentioned above, particular instance section methods were designed with different aims (cleansing, compression), so the relationship between $\mathbf{T}$ and $\mathbf{P}$ can reflect different properties of the training set. In order to validate this relation we examined a set of 11 instance selection methods and analyzed the relationship between compression and prediction accuracy. Having obtained the results we identify which algorithms can be used as landmarks for meta-learning. The study is based on empirical analysis on 80 datasets.

The paper is structured as follows. First, we discuss the state-of-the-art in instance selection and meta-learning, where the most popular algorithms are described in more detail, and then we show and discuss which and why particular evaluated instance selection methods are good landmarks. In Section 5 we perform real-world experiments on meta-learning systems, and finally in Section 6 we summarize the obtained results and draw further research directions.

2. The Instance Selection Methods

The purpose of the instance selection methods is to remove data samples. They take as an input the entire training set $\mathbf{T}$ consisting of n samples ($\mathbf{T} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in \Re^m$, $y_i = \{s_1, s_2, ..., s_c\}$, where s_i denotes i-th symbol), and then eliminate from it useless samples, returning the remaining ones denoted as $\mathbf{P}$. The difference between cardinality of the input dataset n and the output dataset $\|\mathbf{P}\| = n'$, divided by the cardinality of the input dataset n, is called compression.

$$cmp = 1 - \frac{n'}{n} \tag{1}$$

Note that a cmp approaching 1 represents a scenario in which $n' \ll n$, which means many samples are removed from the training set (we would call it high compression as cmp is close to 1) and respectively when cmp is close to 0 or in other words small compression appears when $n' \simeq n$, that means just few instances were removed from $\mathbf{T}$.

In instance selection algorithms, the decision of selection or rejection of particular sample or samples is made by optimizing one of two or both objectives, which are:

- ***Type-I***: Maximization of the accuracy of the classifier. This objective is achieved by eliminating noise samples,
- ***Type-II***: Minimization of the execution time of the classifier. This objective is achieved by reducing the number of reference vectors, and keeping the selected subset of samples as small as possible by rejecting all redundant ones.

Note that this taxonomy is just one possible presentation (a more detailed view can be found in [8]), however it is crucial as these two type of algorithms behave differently considering the relation between compression and prediction accuracy. An illustrative example of the accuracy–compression relation was presented in our preliminary paper [9]. A theoretical background of sampling and instance selection derived from the information theory can be found in [10].

In the paper we use and compare 11 instance selection algorithms, but for 2 of them additional configuration settings are used, so in total we have 13 methods to compare. This set of algorithms includes older ones like the condensed nearest neighbor rule (*CNN*) and the edited nearest neighbor (*ENN*) through to the decremental reduction optimization procedure (*Drop*) family developed at the end of the 1990s, to the more modern algorithms such as the modified selective subset (*MSS*). These are:

- *CNN* [11]. It is an ancestor of all condensation methods, and thus it belongs to the *Type-II* group. It starts by randomly selecting one representative instance per class, and adds it to the reference set $\mathbf{P}$ (the dataset $\mathbf{T}$ remains unchanged while the algorithm works), and then starts the main loop where each misclassified instance from $\mathbf{T}$ by the nearest neighbor classifier trained on $\mathbf{P}$ is added to $\mathbf{P}$. The algorithm stops when all the instances in $\mathbf{T}$ are correctly classified.
- *IB2*. This was developed by Aha et al. as an instance-based learning [12] algorithm (version 2). It is very similar to *CNN*; it also starts by selecting one sample per class, adding it to $\mathbf{P}$, but it only once iterates over all samples in the data, trying to add them to $\mathbf{P}$ if an instance is misclassified. It is also a representative of a *Type-II* family.
- *Gabriel graph editing* (*GGE*) [13]. This method builds a Gabriel graph over training data and then selects border samples and stores them in $\mathbf{P}$. Border samples have at least one instance of another class in their neighborhood according to the Gabriel graph, and therefore this is a *Type-II* method. The Gabriel graph is determined by validating condition

$$\underset{a \neq b \neq c}{\forall} \|\mathbf{x}_a - \mathbf{x}_b\|_2^2 > \|\mathbf{x}_a - \mathbf{x}_c\|_2^2 + \|\mathbf{x}_b - \mathbf{x}_c\|_2^2 \tag{2}$$

between every three instances, where $\{\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c\}$ are instances from $\mathbf{T}$, and $\|\cdot\|_2^2$ denotes the square of the L_2 norm. However, there is is another known version of the Gabriel graph editing method, which keeps the remaining samples in **P** and the border samples are removed. In this case it works as a *Type-I* method, being a regularization for *k*NN. In the experiments we used the first approach, which keeps the border samples only.

- *Relative neighbor graph editing* (*RNGE*) [13]. This method is very similar to *GGE*. The difference is in the graph construction criteria, which is now defined as:

$$\underset{a\neq b\neq c}{\forall} \|\mathbf{x}_a - \mathbf{x}_b\|_2 \geq \max(\|\mathbf{x}_a - \mathbf{x}_c\|_2 + \|\mathbf{x}_b - \mathbf{x}_c\|_2) \tag{3}$$

  As shown in [13], the following relation takes place:

$$\mathbf{T} \supseteq \mathbf{P}_{Voronoi} \supseteq \mathbf{P}_{GGE} \supseteq \mathbf{P}_{RNGE} \tag{4}$$

  where $\mathbf{P}_{Voronoi/GGE/RNG}$ is a set of **P** returned by given instance editing algorithm. The $\mathbf{P}_{Voronoi}$ are border instances according to the Voronoi diagram.
- *Edited nearest neighbor* (*ENN*) [14]. This is another ancestor but for *Type-I* methods, and it is often used as a preprocessing step before other instance selection algorithms. This algorithm analyzes the neighborhood of the given query instance **x**. If this instance is misclassified by its *k* neighbors, it is removed from **P**. Initially $\mathbf{P} = \mathbf{T}$.
- *Repeated ENN* (*RENN*) [15]. This method is an extension of the *ENN* algorithm where the *ENN* algorithm is repeated until no instance is removed (also a *Type-I* method)
- *All-kNN* [15]. This is another extension of the *ENN* algorithm (*Type-I* method) where the *ENN* step is repeated for a range of $k = [k_{min} \ldots k_{max}]$ values.
- *Drop n*. The decremental reduction optimization procedure is a family of five similar algorithms [16], which can be assigned to a mixture of *Type-I* and *Type-II* methods. It can be explained as dropping instances from **P** while at the beginning of the algorithm $\mathbf{P} = \mathbf{T}$. *Drop* algorithms are based on the analysis of the associate array, which is defined as a set of indexes of instances having particular instance $\mathbf{x}_i$ in its neighborhood. In *Drop* 1 the algorithm analyzes if the removal of instance **x** from the associates does not affect the classification accuracy, if so **x** is then removed. The *Drop* 5 algorithm is similar to the *Drop* 2 algorithm (thus only one was used in the experiments); again it analyzes the effect of the removal of $\mathbf{x}_i$ from the associates array, but in the analysis the order of removal becomes crucial as instances are ordered according to the distance to the so-called nearest enemy (the nearest vector from the opposite class) starting with the farthest instances first. The *Drop* 3 and *Drop* 4 algorithms are similar to *Drop* 1 but before selection the *ENN* algorithm is used to prune the dataset. These last two versions have not been used in the experiments; instead *Drop* 1 and *Drop* 5 were analyzed.
- *Iterative case filtering* (*ICF*) [17]. This is a two-step algorithm. First, it applies the *ENN* algorithm to prune noisy samples, then in the second step it finds a *local set* $\phi(\mathbf{x})$ for every instance $\mathbf{x}_i$ ($\phi(\mathbf{x}_i)$), defined by the largest hypersphere centered at $\mathbf{x}_i$ which includes only instances of the same class as $\mathbf{x}_i$. The *local set* is then used to calculate two statistics:

$$Coverage(\mathbf{x}) = \left\{\mathbf{x}' \in \mathbf{T} : \mathbf{x} \in \phi(\mathbf{x}')\right\} \tag{5}$$

$$Reachability(\mathbf{x}) = \left\{\mathbf{x}' \in \mathbf{T} : \mathbf{x}' \in \phi(\mathbf{x})\right\} \tag{6}$$

  and if $reachability(\mathbf{x}_i) > coverage(\mathbf{x}_i)$ then the sample $\mathbf{x}_i$ is removed from **P**. *ICF* is another example of a mixed *Type-I* and *Type-II* method.
- *Modified selective subset* (*MSS*) [18]. This is a modification of the selective subset algorithm proposed by Ritter [19]. In the basic algorithm the authors define the so-called selective subset as a subset **P**, which is consistent (all samples in the training set **T** are correctly classified by the nearest neighbor rule based on **P**). Samples from the training set **T** must be closer to instances in **P** from the same class, and **P** must be as small as possible. The modified version applies changes to the definition of the selective subset, and selects examples which are closest to the nearest enemy of given sample. It belongs to the *Type-II* family

A common strategy in the development of more recent instance selection algorithms is the assembly of both objectives into one algorithm, for example as in *noise removing based on minimal consistent set* (NRMCS) [20] or *class conditional instance selection* (CCIS) [21], or even in the *Drop*s or *ICF*, where first the noise filter is applied and then the condensation step starts. This usually results in a better and more consistent set of reference vectors being obtained, but as we will further show, these mixed type (*Type-I* and *Type-II*) methods negatively affect the relation between compression and prediction accuracy. For more details of classical instance selection methods and their comparison readers are referred to [8,22–24].

Instance selection methods are still under rapid development. Recent methods in the field first solve other problems and then use classification, namely, regression problems as in [25–30], instance selection in data streams as in [31–33], and time series classification [34,35], or build ensembles of instance selection [36–40] and even create meta-learning systems, which automatically adjust a proper instance selection method to a given dataset as in [41,42].

3. Meta-Learning

Meta-learning is the efficient selection of the best prediction model for a given training set **T**. In other words, it is a recommendation system which recommends a prediction model for a given training set. To achieve this goal, instead of validating each prediction model $M()$ (*base model*) on training set **T**, meta-data descriptors are extracted from **T**, so that the training set is represented as a single instance of meta-data features and used by another previously trained meta-model (usually a classical regression model) which predicts the quality of $M()$ (the *meta-model application* frame in Figure 2). This significantly reduces the computational complexity, because to select the best model we do not need to train and optimize parameters of each $M()$, and instead estimate its accuracy by using meta-data and the meta-model [43]. Each meta-model is trained for a single base model using a meta-set (the *meta-model training* frame in Figure 2). Finally, the meta-set is obtained out of collection of historical datasets (a repository) for which we already know both the meta-date descriptors and the performance of the base model, so that each instance of the meta-set consists of both the meta-data extracted from given dataset, and a label representing highest accuracy of the base model (see the *meta-set construction* frame in Figure 2). In applications we need several meta-models, each dedicated to a specific base model, for example one for the support vector machine (SVM), one for *k*NN, etc. The obtained estimated performances of the base models are then ranked and the best one is selected for final application. Another common approach replaces the regression-based meta-model with the *k*NN model, which identifies the most similar dataset from the metaset (the nearest neighbors according to the meta-feature space), and returns an aggregated ranking of the best models obtained for each of the most similar datasets [44]. In the experiments we follow the first approach.

In both approaches the quality of the meta-model strongly depends on selected meta-data descriptors also called meta-features, which should reflect the prediction power of given base model. Note that we need a separate meta-model for each base model. An extension of this approach even allows for the estimation of the value of hyper-parameters [45] of the base model or at least helps to decide whether hyper-parameter optimization is needed [46].

The problem is finding good meta-data descriptors which would allow us to accurately estimate the accuracy or other desired values like execution or training time, etc. The basic examples of meta-learning can be found in the *variable bias management system* (VBMS) [47] which automatically choses a learning algorithm based on two meta-parameters: the number of features and the number of vectors, often extended by descriptors representing the number of features of given type.

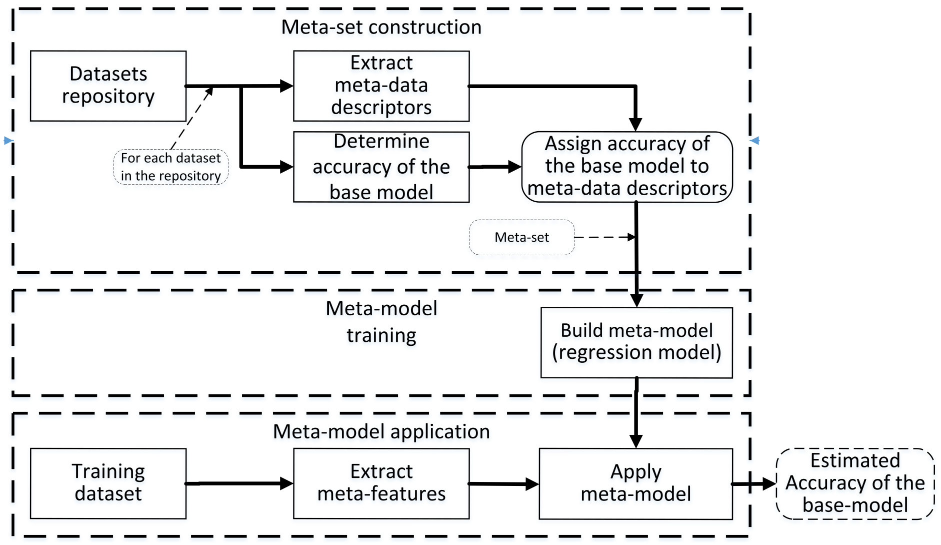

Meta-set construction
Datasets repository
Extract meta-data descriptors
Determine accuracy of the base model
Assign accuracy of the base model to meta-data descriptors
For each dataset in the repository
Meta-set
Meta-model training
Build meta-model (regression model)
Meta-model application
Training dataset
Extract meta-features
Apply meta-model
Estimated Accuracy of the base-model

Figure 2. The process of construction of a meta-learning system.

An extension of this idea is a meta-learning system which is based on a set of meta-variables describing aggregated statistical and information theory properties of individual attributes and class labels [5,6,48–50]. Commonly reported meta-features of this type include: canonical correlation for the best single combination of features (called *cancor1*), canonical correlation for the best single combination of features orthogonal to *cancor1*, the first normalized eigenvalue of the canonical discriminant matrix, the second normalized eigenvalue of the canonical discriminant matrix, the mean kurtosis of attributes of **T**, the mean skewness of attributes of **T**, the mean mutual information of class and feature, joint entropy of a class variable and attribute, and entropy of classes, etc.

Another approach to determine valuable meta-data descriptors is the idea of landmarking [7,51]. It utilizes the relationship between the accuracies obtained by simple predictors with low computational complexity such as naive Bayes, 1NN, decision tree, and complex data mining algorithms like SVM, neural networks, or other systems. The selection of meta-features describing the dataset is the real challenge. Various authors use different combinations of landmarks and not only provide performance of the simple classifiers but also use model-based features where the structure of a simple model is provided as a dataset descriptor. A common solution is based on unpruned decision tree properties as described in [52]. In this case, to represent the complexity of the data, authors suggested the ratio of the number of tree nodes to the number of attributes, the ratio of the number of tree nodes to the number of training instances, the average strength of support of each tree leaf, the difference in the gain-ratio between the attributes at the first splitting point of the tree building process, maximum depth, number of repeated nodes, a function of the probabilities of arriving at the various leaves given a random walk down the tree, the number of leaves divided by tree shape, and the number of identical multi-node subtrees repeated in the tree.

In [53] the author studied a set of data complexity measures for classification problems to assess the accuracy. He divided these measures into three groups: Group 1 is comprised of measures of overlaps in feature values from different classes like the *maximum Fisher's discriminant ratio* and the *maximal (individual) feature efficiency*; Group 2 refers to measures of separability of classes which include the *minimized sum of error distance by linear programming*, the *error rate of linear classifier by linear programming*, the *fraction of points on class boundary*, the *fraction of points on the class boundary*, and the *error rate of the 1 nearest neighbor classifier*; and Group 3 contains measures of geometry, topology, and density of manifolds containing *nonlineality of linear classifiers by linear programming* and the *fraction of points with associated adherence subsets retained*. These measures are then compared using three artificially designed datasets and three classifiers: *k*NN, C4.5, and SVM, and indicate which measure is suitable to which classifier. Recently, in [54] authors suggested the generalization of meta-feature construction by defining a framework which covers all of already known meta-descriptors. It is based on defining three elements of the so-called objects: the source of meta-data (e.g., the dataset, the simple model, etc.); the meta-function (the element extracted from the object); and a post processing operation (the aggregation function). An integrated approach which joins all popular types of meta-features is available in the *Pattern Recognition Engineering* (PaREn) system [55] (recently renamed to MLWizard). It was used in the experiments discussed in Section 5 as a reference solution (the list of meta-features extracted by PaREn is available in Table 1).

Table 1. Description of the meta-features extracted by the PaREn meta-learning system. The *aggregation function* column describes the function used to aggregate a list of values. If multiple functions are provided, the final metaset contains a separate feature for each aggregation function.

Description	**Aggregation Function**
The number of samples	none
The number of attributes	none
The number of classes	none
The number of numerical attributes	none
The number of nominal attributes	none
The ratio of the number of numerical features to the number of all features	none
The ratio of the number of nominal features to the number of all features	none
The ratio of the number of samples to the number of attributes	none
The collection of the number of symbols in nominal attributes	min, max, mean, deviation
The collection of entropy of each attribute	min, max, mean, deviation
The collection of conditional entropies for each attribute	min, max, mean, deviation
The collection of the mutual informations for each attribute	min, max, mean, deviation
The class label entropy	none
The ratio of the class entropy and the average mutual information	none
The noise to signal ratio: the difference between mean entropy and mean mutual information divided by the mean mutual information	none
The kurtosis of attributes	min, max, mean, deviation
The skewness of the attributes	min, max, mean, deviation
The landmark: the naive-Bayes performance	none
The landmark: the *1*NN performance	none
The landmark: the accuracy of the DecisionStump model trained on the best attribute according to information gain criteria	none
The landmark: the average accuracy of the DesisionStump models trained on a single attribute	none
The landmark: the accuracy of the DecisionStump model trained on the worst attribute according to information gain criteria	none
The landmark: the number of nodes	none
The landmark: the number of leaves in an unpruned tree	none
The landmark: the collection of lengths of the tree branches	min, max, mean, deviation
The landmark: the collection of the number of branches which reached a given depth	min, max, mean, deviation
The landmark: the tribute usage statistics (counts how often each attribute was used to construct the tree)	min, max, mean, deviation

In many cases we are not exactly interested in estimating prediction accuracy but rather in ranking the top classifiers. This problem was studied by Brazdil et al. [44] who suggested use of a *k*NN algorithm to naturally rank results according to distances in the meta-feature space. This approach was further extended. For example, Sun and Pfahringer proposed the forest of the approximate ranking tree which is trained on pairwise comparisons of the base models and the metaset [56].

An alternative approach to meta-learning was proposed by Grąbczewski and Jankowski in [57,58]. It focuses on so called machine unification, such that the already-trained machines (a machine is a prediction model or data transformer) are cached and re-used when needed. This concept allows us to cache parts of the models and utilize them inside other complex machines. The proposed solution also ranks models according to the execution time such that the most promising machines are evaluated first. Note that this concept applies to both stages (2) and (3) of the data mining process.

For more details on meta-learning methods the readers are referred to [59].

4. The Relationship between Dataset Compression and Prediction Accuracy

In Section 2 we distinguished two basic types of instance selection objectives called *Type-I* and *Type-II* and also defined a third group constituting a mixture of these two. These types are important because the relation between compression and accuracy behaves differently. Considering *Type-I*, for *k*NN regularization methods, intuitively, when the dataset is clean without any noisy samples or outliers there is no need to remove any instances. The heuristics built into the algorithm may treat only some of the border instances as noise and remove them. As a result the compression is low. When the level of noise in the data increases (for example by mislabeling) or the decision boundary becomes jagged, the number of instances which are removed increases, and the compression grows. Now considering the prediction system, when the dataset is clean and the decision boundary is smooth and simple then the classification accuracy will be high. In the opposite situation the classification accuracy will drop, therefore intuitively we expect these two values (compression and accuracy) to remain in an inverse relation.

For *Type-II* methods, which are aimed at compacting training data, when the decision boundary is smooth the dataset can be easily compacted and many instances can be removed, so the compression is then high. The noise in the data or a complex and jagged decision boundary reduce the compression as there are less regularities. In this case we expect an inverse relation in comparison to *Type-I* methods, so increasing compression should reflect the increase in prediction accuracy. In [9] we have evaluated this relation empirically on artificial and real-world datasets for *k*NN classifier and two methods: *CNN* and *ENN*. Now we want to extend this work, and evaluate other more advanced instance selection methods which use more advanced heuristics to determine the final set **P** as discussed in the Section 2. Note that some of the methods, such as the Drop family or ICF, combine both *Type-I* and *Type-II*. That means that the resulting compression rates may incorrectly reflect the complexity of the data set, since the relation between compression and accuracy for *Type-I* and *Type-II* remains in contradiction.

In order to solve this problem, we carried out a number of experiments on real-world datasets to verify which instance selection methods preserve strong correlation between compression and prediction accuracy. Because in instance selection applications two scenarios are possible (one in which the prediction model is trained on the entire training set without compression, and one in which the dataset used to train the predictor is already filtered by the instance selection), we evaluate both of them, and call them respectively *Case A* and *Case B*.

4.1. The Experimental Setup

Both experiments were carried out on 80 real-world datasets obtained from the Keel repository [60] and UCI repository [61]

Note that some of the datasets in both repositories have the same name but they have different content (different number of attributes or different number of instances). The *Src* column in Table 2 indicates the source repository of given dataset, and the datasets are also available in the supplementary materials. In the study we compare 11 instance selection methods described in Section 2 and for *Drop* methods we also evaluated two additional parameters resulting in a total of 13 methods. The datasets used in the experiments represent a broad spectrum of classification problems from small ones such as *appendicitis* which has 106 samples, up to larger ones such as the *shuttle* dataset which has almost 58,000 samples. The datasets also contains different types of attributes, from simple, all-numerical features, through to mixed type features, to all-nominal features. The basic information about these datasets is provided in Table 2, where the column *Src* indicates the source of the dataset.

For all of the symbolic attributes in any dataset, the Value Difference Metric (VDM) [62] was used, or more precisely, the heterogeneous version HVDM [63], but instead of direct application of VDM metric each symbolic attribute a was encoded on c attributes (c is the number class labels), where each new attribute represents conditional probability that the output class $y = s_j, (s \in s_1, s_2, ..., s_c)$ given attribute a has value v_i, where v_i is one of the symbols of attribute a. As shown in [64], this type of conversion is equivalent to the HVDM metric and is more accurate than the binary coding of symbolic attributes (the Hamming distance). After the feature type conversion, all attributes were normalized to the $[0, 1]$ range and then the 10-fold cross-validation test was used to evaluate the prediction accuracy.

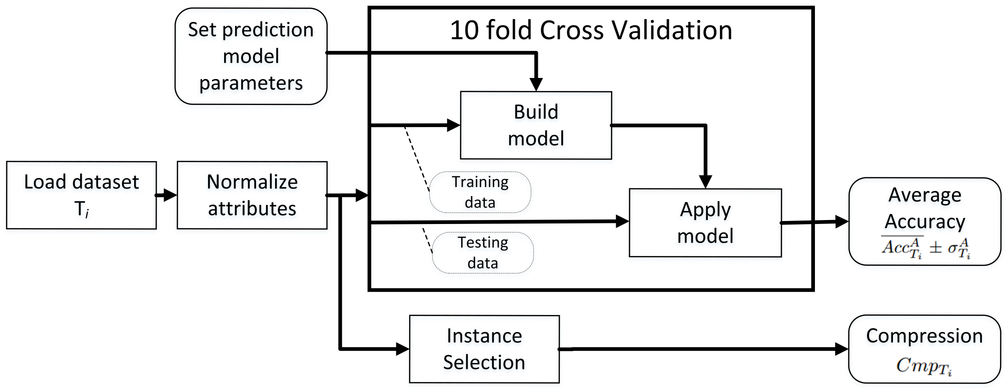
Earlier it was mentioned that the experiments were split into two scenarios. In the first one, denoted as *Case A* presented in Figure 3a, the instance selection was performed outside the cross-validation, and the prediction model was built on an uncompressed dataset such that instance selection did not affect the training set. This experiment mimics a real use-case where instance selection is executed once to estimate compression to assess the dataset properties and the cross-validation procedure is used to estimate the prediction accuracy of a given base model—a typical meta-learning scenario. Note that in all experiments hyper-parameters of instance selection were not optimized, and only the default values were used (see Table 3), but the hyper-parameters of the prediction model were optimized to maximize prediction accuracy. For each setting of the classifier an independent model was built and only the best obtained results were recorded for the investigation.

Table 2. Datasets used in the experiments with basic statistics. Column *Src* indicates source of the dataset, where *K* denotes the Keel project and *U* denotes the UCI repository.

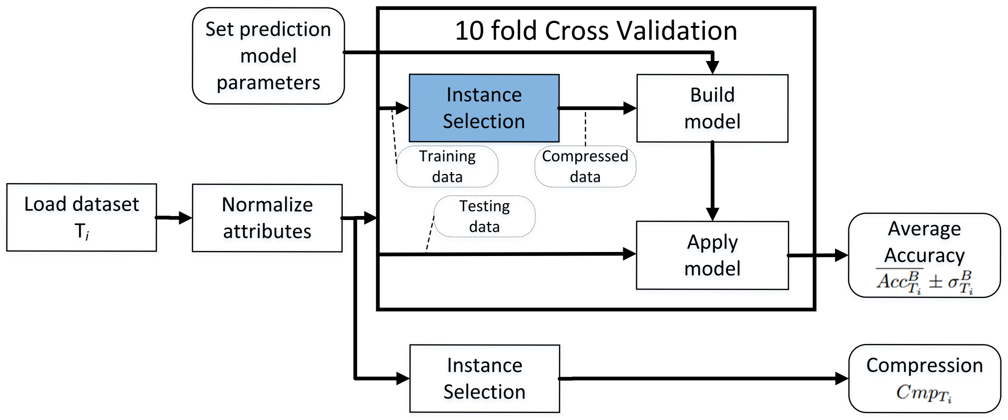
ID	Dataset (T)	*Src*	Samples	Attr	Classes
1	anneal	*U*	898	38	5
2	appendicitis	*K*	106	7	2
3	balance	*K*	625	4	3
4	banana	*K*	5300	2	2
5	bands	*K*	365	19	2
6	bupa	*K*	345	6	2
7	car	*U*	1728	6	4
8	cleveland	*K*	297	13	5
9	cmc	*U*	1473	9	3
10	coil2000	*K*	9822	85	2
11	ecoli	*U*	336	7	8
12	flags	*U*	194	29	194
13	glass	*U*	214	9	6
14	glass	*K*	214	9	6
15	haberman	*U*	306	3	2
16	haberman	*K*	306	3	2
17	hayes-roth	*K*	160	4	3
18	heart	*K*	270	13	2
19	hepatitis	*K*	80	19	2
20	ionosphere	*K*	351	33	2
21	iris	*K*	150	4	3
22	kr-vs-kp	*U*	3196	36	2
23	led7digit	*K*	500	7	10
24	letter	*K*	20,000	16	26
25	liver-disorders	*U*	345	6	2
26	lymph	*U*	148	18	4
27	magic	*K*	19,020	10	2
28	mammographic	*K*	830	5	2
29	marketing	*K*	6876	13	9
30	mfeat-factors	*U*	2000	216	10
31	mfeat-fourier	*U*	2000	76	10
32	mfeat-karhunen	*U*	2000	64	10
33	mfeat-morphological	*U*	2000	6	10
34	mfeat-pixel	*U*	2000	240	10
35	mfeat-zernike	*U*	2000	47	10
36	promoters	*U*	106	58	2
37	monk-2	*K*	432	6	2
38	monks-problems-1	*U*	556	6	2
39	monks-problems-3	*U*	554	6	2
40	movement_libras	*K*	360	90	15
41	newthyroid	*K*	215	5	3
42	optdigits	*K*	5620	64	10
43	page-blocks	*K*	5472	10	5
44	penbased	*K*	10,992	16	10
45	phoneme	*K*	5404	5	2
46	pima	*K*	768	8	2
47	ring	*K*	7400	20	2
48	satimage	*K*	6435	36	6
49	segment	*K*	2310	19	7
50	shuttle	*K*	57,999	9	7
51	sonar	*K*	208	60	2
52	spambase	*K*	4597	57	2
53	spectfheart	*K*	267	44	2
54	spectrometer	*U*	531	102	48
55	tae	*K*	151	5	3
56	texture	*K*	5500	40	11
57	thyroid	*K*	7200	21	3
58	tic-tac-toe	*U*	958	9	2
59	titanic	*K*	2201	3	2
60	twonorm	*K*	7400	20	2
61	vehicle	*K*	846	18	4
62	vowel	*K*	990	13	11
63	waveform-5000	*U*	5000	40	3
64	wdbc	*K*	569	30	2
65	wine	*K*	178	13	3
66	wisconsin	*K*	683	9	2
67	yeast	*K*	1484	8	10
68	abalone	*K*	4174	8	28
69	automobile	*K*	159	25	6
70	breast	*K*	277	9	2
71	chess	*K*	3196	36	2
72	crx	*K*	653	15	2
73	flare	*K*	1066	11	6
74	german	*K*	1000	20	2
75	housevotes	*K*	232	16	2
76	mushroom	*K*	5644	22	2
77	nursery	*K*	12,960	8	5
78	saheart	*K*	462	9	2
79	splice	*K*	3190	60	3
80	zoo	*K*	101	16	7

The second experiment, called *Case B* (presented in Figure 3b), was aimed at repeating the previous one but in this case the prediction model was built using training set filtered out by the instance selection method. This experiment mimics a typical instance selection use-case where the prediction model is built on the dataset initially compressed by the instance selection algorithm. Again, hyper-parameters of instance selection were not tuned; only the prediction model was optimized to achieve the highest accuracy.

In all the conducted experiments all datasets were initially randomly divided into 10 subsets (preserving class frequency—a so-called stratified sampling) and these subsets were used in the cross-validation test.



(a) *Case A*



(b) *Case B*

Figure 3. Two experimental scenarios were used to analyze the *compression–accuracy* relationship for each individual instance selection method and a given prediction model. The first scenario (**a**), represents situation where the training model was constructed using entire training set available in given validation. In the scenario (**b**), the training set was first filtered out using the instance selection method.

Table 3. The parameter settings of the instance selection methods. *CNN*: condensed nearest neighbor rule; *GGE*: Gabriel graph editing; *RNGE*: relative neighbor graph editing; *ENN*: edited nearest neighbor; *RENN*: repeated ENN; *ICF*: iterative case filtering; *MSS*: modified selective subset; *IB2*: instance-based learning type 2; *Drop*: decremental reduction optimization procedure.

Instance Selection	Parameter	Range
CNN	k	1
IB2	k	1
GGE	k	1
RNGE	k	1
ENN	k	3
RENN	k	3
All-kNN	k_{min}	3
	k_{max}	5
Drop 1 (k1)	k	1
Drop 1 (k3)	k	3
Drop 5 (k1)	k	1
Drop 5 (k3)	k	3
ICF	k	3
MSS	k	1

In the experiments the state-of-the-art and the most robust and commonly used prediction models were used, such as SVM [65–67] and random forest [68–70], as well as the *k*NN model. All the experiments were executed using RapidMiner software as a shell system [71]. Instance selection was performed with the *Information Selection extension* [72] developed by the author, which includes the *Instance Selection Weka plug-in* created by Arnaiz-González and García-Osorio [26]. The SVM implementation was based on LibSVM library, and random forest was used from the Weka suite. For each prediction model the meta parameters such as k for *k*NN, C and γ for SVM, and number of trees of random forest were optimized using the greed procedure. In the final results only the highest average accuracy was reported. Hyperparameters evaluated in the experiments are shown in Table 4, and RapidMiner processes representing the experiments are available in supplementary materials.

Table 4. The parameter settings of the prediction methods.

Model	Parameter	Range
*k*NN	k	$[1, 2, \ldots, 40]$
SVM	C	$[0.001, 0.01, 0.1, 1, 10, 100]$
	γ	$[0.05, 0.1, \ldots, 0.6]$
Random Forest	# trees	$[10, 20, 30, \ldots, 100]$

4.2. Compression-Accuracy Relation for the Case A Scenario

As stated in the previous section, *Case A* assumes that compression and prediction accuracy are evaluated independently, and the goal of the experiments is to verify a typical meta-learning use-case where compression is used as a landmark and does not affect the training set. For that purpose for each of the datasets presented in Table 2, compression as well as prediction accuracy were measured (labeled respectively as Cmp_{T_i} and $\overline{Acc^A_{T_i}} \pm \sigma^A_{T_i}$, where T_i denotes i'th dataset, and the A superscript is used to distinguish the values recorded in *Case A*). The experimental process is described in the scheme presented in Figure 3a and the results were collected for all instance selection methods and the all above-described classifiers.

To measure the relation between Cmp and Acc we used three different measures. These were the Pearsons correlation coefficient $r(Cmp, \overline{Acc^A})$, root- mean-square error (RMSE) of the linear model, and the coefficient of determination (R^2) of the same model. The correlation measure was evaluated with a confidence level of 0.05, and results are shown in Table 5.

Table 5. The relationship between compression and classifier accuracy for different instance selection methods and classifiers. The relationship is measured using Pearsons linear correlation ($r(\cdot)$), for which both the value of correlation (corr.) and p-value are provided, the root-mean-square error (RMSE), and the coefficient of determination (R^2) of linear regression model for the *Case A* scenario. The best results for each classifier are marked in bold.

Method	***k*NN**				**SVM**				**Random Forest**			
	$r\,(Cmp, Acc)$		**RMSE**	R^2	$r\,(Cmp, Acc)$		**RMSE**	R^2	$r\,(Cmp, Acc)$		**RMSE**	R^2
	Corr.	***p*-Value**			**Corr.**	***p*-Value**			**Corr.**	***p*-Value**		
CNN	0.937	0.000	0.051	0.878	0.902	0.000	0.063	0.814	0.900	0.000	0.063	0.810
ENN	−0.965	0.000	0.038	0.930	−**0.924**	0.000	**0.056**	**0.854**	−0.917	0.000	0.058	0.842
RENN	−**0.974**	0.000	**0.033**	**0.949**	−0.912	0.000	0.060	0.832	−0.904	0.000	0.062	0.817
All-kNN	−0.885	0.000	0.068	0.783	−0.841	0.000	0.079	0.707	−0.857	0.000	0.075	0.734
IB2	0.955	0.000	0.043	0.912	0.916	0.000	0.058	0.839	**0.920**	0.000	**0.057**	**0.847**
GGE	0.333	0.003	0.137	0.111	0.232	0.040	0.142	0.054	0.288	0.010	0.139	0.083
RNGE	0.634	0.000	0.112	0.403	0.560	0.000	0.120	0.313	0.586	0.000	0.118	0.344
Drop 1 ($k = 3$)	0.520	0.000	0.124	0.271	0.459	0.000	0.129	0.210	0.481	0.000	0.128	0.231
Drop 1 ($k = 1$)	0.463	0.000	0.129	0.215	0.421	0.000	0.132	0.177	0.444	0.000	0.130	0.197
Drop 5 ($k = 3$)	0.550	0.000	0.122	0.302	0.486	0.000	0.127	0.236	0.488	0.000	0.127	0.238
Drop 5 ($k = 1$)	0.526	0.000	0.124	0.277	0.460	0.000	0.129	0.212	0.475	0.000	0.128	0.226
ICF	−0.023	0.838	0.145	0.001	−0.043	0.703	0.145	0.002	0.019	0.869	0.145	0.000
MSS	0.787	0.000	0.090	0.620	0.718	0.000	0.101	0.516	0.721	0.000	0.101	0.520

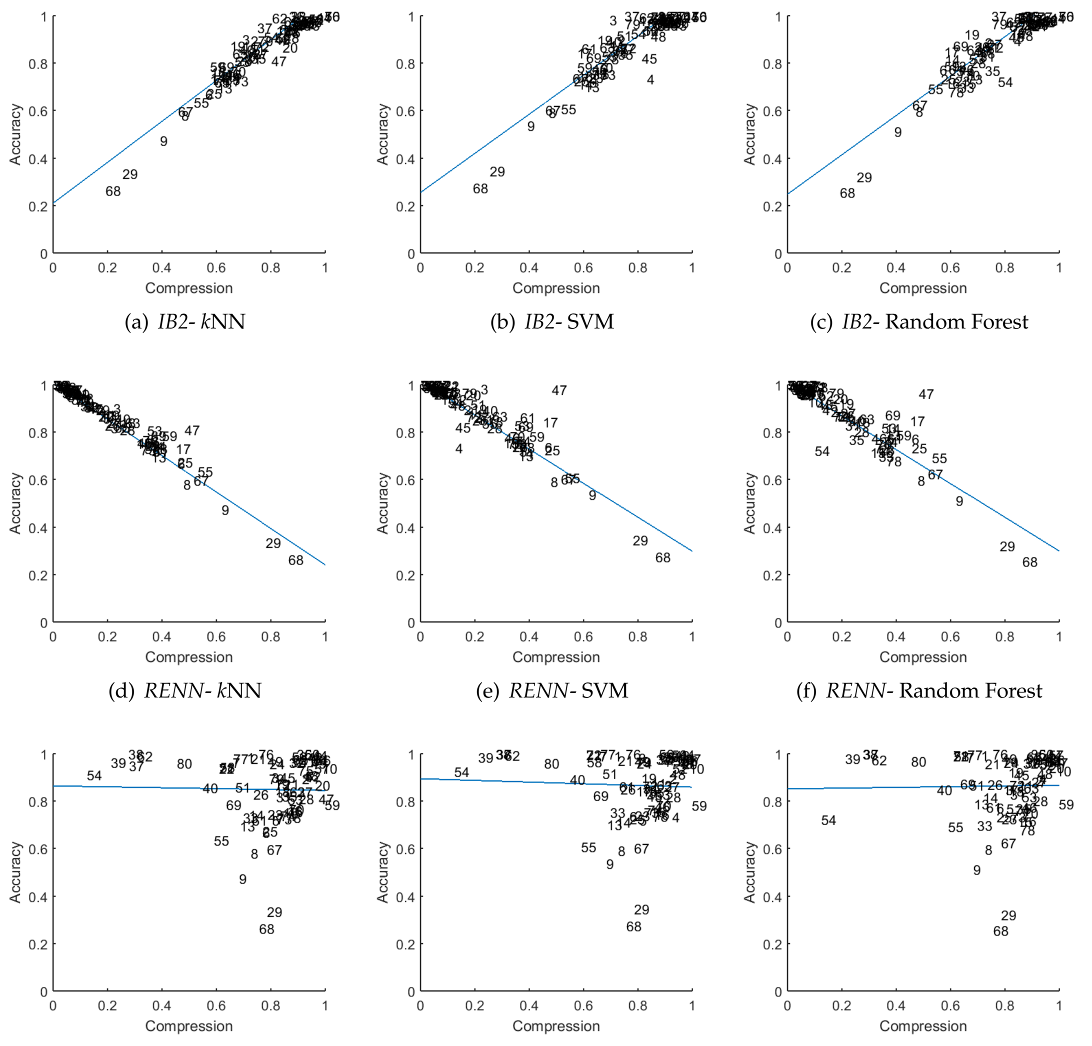
These results indicate that compression is significantly correlated with accuracy, at least for the majority of the analyzed compression methods. For the four methods *CNN*, *IB2*, *ENN*, and *RENN* the absolute value of correlation for all classifiers is above 0.9. For one it is almost 0.9 (the *All-kNN*), and for the remaining ones it is above 0.5. The exception is *ICF*, for which the correlation is very low and close to 0. The *ENN*, *RENN*, *All-kNN*, and *ICF* have negative values of correlation coefficients, and this is consistent with the above-described interpretation of *Type-I* and *Type-II* methods.

Likewise, the other measures show similar behavior, although the RMSE between the true and the estimated accuracy clearly indicates that for half of the instance selection methods (*Drop*, *RNGE*, *GGE* and *ICF*) the error is almost four times higher compared to the best result. Also noticeable is the difference between the RMSE obtained when estimating the accuracy of the *k*NN classifier and the remaining base-models, where the $\mathrm{RMSE}^{k\mathrm{NN}}$ is approximately 1.7 times smaller than the $\mathrm{RMSE}^{\mathrm{SVM}}$ and $\mathrm{RMSE}^{\mathrm{Random\ Forest}}$. Also, the R^2 measure drops by 10% between best results obtained for *k*NN and the other classifiers. This is reasonable as instance selection methods internally use *k*NN to determine which instances should be removed or kept, so that it better reflects the performance of *k*NN classifier.

Of the collected results for three instance selection methods: *RENN*, *IB2* and *ICF*, we drew figures representing the Cmp_T-Acc_T relation (see Figure 4). These methods represent the highest correlation for the *Type-I* method, the highest correlation for the *Type-II* method, and the worst results. In the figures the X-axis represents Cmp_T and the Y-axis represents Acc_T; each data-point is marked with a number i and represents a pair of $< Cmp_{T_i}, \overline{Acc^A_{T_i}} >$ obtained for i-th dataset (see ID column in Table 2). The blue line shows the linear regression (LR) model $Acc = a \cdot Cmp + b$ for which the RMSE and R^2 were measured.

The first two rows of Figure 4 confirm a strong linear relation between accuracy and compression. Interestingly, comparing results of *IB2* and *RENN* for SVM (but also for random forest) some of the outlying data points represent different datasets. For example, in Figure 4e the most outlying dataset is number 47 which is not easy to recognize (it is almost in line with the others) in Figure 4b. This property is desired, as compression of these two methods partially complements each other.

The third row shows the *ICF* compression in relation to the performance of the classifiers. Here we can not see any relation; the data points are randomly distributed around (1,1) coordinates. For some datasets which have a very low compression we observe a very high performance (for example for the *spectrometer* dataset ($id = 54$)). Also, for datasets for which compression is high, the performance is very low (as for the *marketing* ($id = 29$) or *abalon* datasets (ID= 68)). The explanation of these results will be presented in Section 4.4.



(a) *IB2*- *k*NN (b) *IB2*- SVM (c) *IB2*- Random Forest

(d) *RENN*- *k*NN (e) *RENN*- SVM (f) *RENN*- Random Forest

(g) *ICF*- *k*NN (h) *ICF*- SVM (i) *ICF*- Random Forest

Figure 4. Relation between compression and prediction accuracy for three selected instance selection methods and three classifiers for the *Case A* scenario. Rows represent three instance selection methods, respectively *ENN* (the best *Type-I* method), *CNN* (the best *Type-II* method), and *ICF* (the method for which the worst results were obtained). Columns represent three classifiers, respectively *k*NN, SVM, and random forest.

4.3. Compression–Accuracy Relation for the Case B Scenario

The *Case B* experiment (Figure 3b) was designed to evaluate the compression–accuracy relation for scenario where the prediction model is built on a compressed dataset. In this case the experiment returned an average accuracy denoted as $\overline{Acc^B_{T_i}} \pm \sigma^B_{T_i}$ (here we use the B superscript to distinguish accuracy obtained in this scenarios) and compression Cmp_{T_i}. The results are presented in the identical form to *Case A* and the same quality measures are used (see Table 6). The main difference is in the accuracy of classifiers. The RMSE, R^2, and the correlation r were calculated between $\langle Cmp_{T_i}, \overline{Acc^B_{T_i}} \rangle$. Note that in *Case B* we used compression identical to that used in *Case A* (calculated in *Case A*, hence without cross-validation), to make the results of *Case A* and *Case B* comparable.

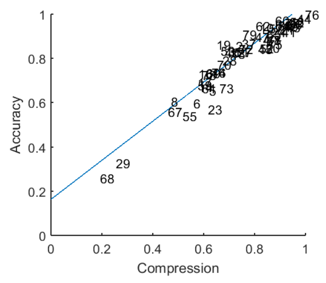
Table 6. Relation between compression and classifier accuracy for different instance selection methods and classifiers. The relation is measured using the Pearson linear correlation ($r(\cdot)$), for which both the value of correlation (corr.) and p-value are provided, the RMSE error, and the coefficient of determination (R^2) of the linear regression model for the *Case B* scenario. The nest results for each classifier are marked in bold.

Method	***k*NN**				**SVM**				**Random Forest**			
	$r\,(Cmp, Acc)$		**RMSE**	R^2	$r\,(Cmp, Acc)$		**RMSE**	R^2	$r\,(Cmp, Acc)$		**RMSE**	R^2
	Corr.	***p*-Value**			**Corr.**	***p*-Value**			**Corr.**	***p*-Value**		
CNN	0.939	0.000	0.053	0.882	0.859	0.000	0.081	0.738	0.883	0.000	0.077	0.780
ENN	−0.949	0.000	0.051	0.900	−0.927	0.000	0.059	0.859	−0.937	0.000	0.054	0.878
RENN	−**0.966**	0.000	**0.044**	**0.934**	−**0.953**	0.000	**0.050**	**0.908**	−**0.974**	0.000	**0.037**	**0.948**
All-kNN	−0.909	0.000	0.068	0.826	−0.893	0.000	0.072	0.798	−0.904	0.000	0.068	0.816
IB2	0.950	0.000	0.048	0.903	0.871	0.000	0.078	0.758	0.875	0.000	0.080	0.766
GGE	0.430	0.001	0.138	0.185	0.255	0.058	0.153	0.065	0.374	0.005	0.144	0.140
RNGE	0.657	0.000	0.113	0.431	0.562	0.000	0.131	0.316	0.615	0.000	0.122	0.378
Drop 1 ($k = 3$)	0.526	0.000	0.139	0.277	0.419	0.001	0.154	0.175	0.388	0.003	0.156	0.150
Drop 1 ($k = 1$)	0.463	0.000	0.136	0.214	0.407	0.002	0.148	0.165	0.317	0.016	0.155	0.100
Drop 5 ($k = 3$)	0.580	0.000	0.133	0.336	0.512	0.000	0.137	0.262	0.465	0.000	0.143	0.216
Drop 5 ($k = 1$)	0.506	0.000	0.140	0.256	0.430	0.001	0.149	0.185	0.442	0.001	0.142	0.195
ICF	0.063	0.641	0.170	0.004	0.017	0.898	0.172	0.000	−0.014	0.918	0.172	0.000
MSS	0.569	0.000	0.141	0.324	0.520	0.000	0.147	0.271	0.561	0.000	0.144	0.315

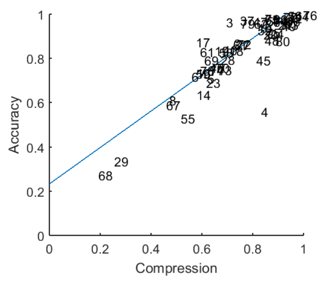
Again, the obtained results indicate a significant relationship between compression and accuracy, but not in all of the cases, as in *Case A*. The strongest relationship was observed for the same methods, which are: *CNN*, *IB2*, *ENN*, *RENN*, and *All-kNN*. The other methods have also similar values; only the correlation sign of *ICF* changed, but this may be a result of numerical artifacts because the value is also close to 0. The plots are also presented for *RENN*, *IB2*, and *ICF* to make them comparable to the previous results. Obtained results are shown in Figure 5, where each row represents an individual instance selection algorithm, columns represent different classifiers, and the blue line in each plot is the linear regression model.

The plots are very similar to those of *Case A*. Again, the compression of *Type-I* (*ENN*, *RENN*, *All-kNN*) methods shows a linear relation to the accuracy, and increasing compression leads to a drop in accuracy. For *Type-II* methods (*CNN*, *IB2*, *GGE*, *RNGE*, *MSS*), an increase of compression correlates with an increase of accuracy. The mixture methods such as the *Drop* family or *ICF* do both, and display low correlation.

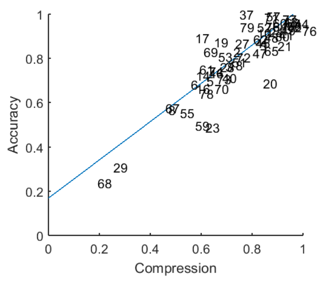
Comparing results from *Case A* and *Case B*, especially for the SVM and random forest, in *Case B* we observe a smaller RMSE, and higher R^2 and correlation values. This results from the dataset compression process which adjusted the datasets to the *k*NN classifier. For example, this can be observed for the dataset with ID 47, which is in line in Figure 5e of *Case B*, while in *Case A* (Figure 4e) it significantly increases the error.



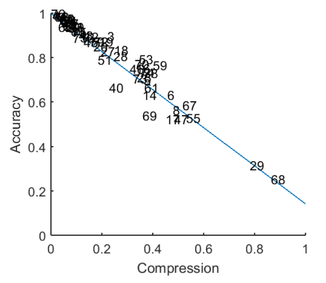
(a) *IB2*- *k*NN



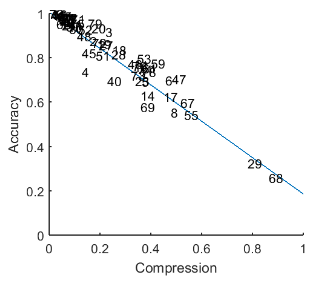
(b) *IB2*- SVM



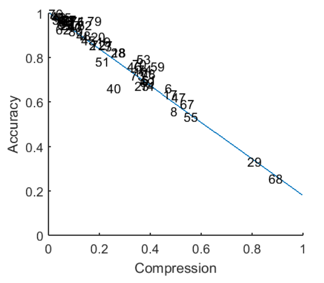
(c) *IB2*- Random Forest



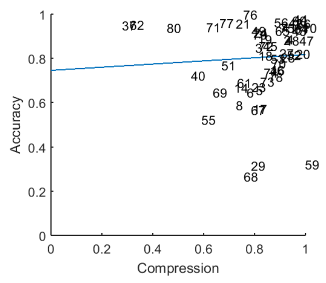
(d) *RENN*- *k*NN



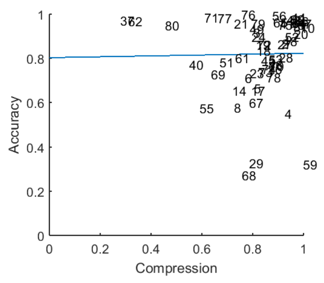
(e) *RENN*- SVM



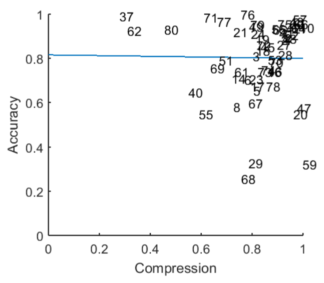
(f) *RENN*- Random Forest



(g) *ICF*- *k*NN



(h) *ICF*- SVM



(i) *ICF*- Random Forest

Figure 5. Relation between compression and prediction accuracy for three selected instance selection methods and three classifiers for the *Case B* scenario. Rows represent three instance selection methods, respectively *ENN* (the best *Type-I* method), *CNN* (the best *Type-II* method), and *ICF* (the method for which the worst results were obtained). Columns represent three classifiers, respectively *k*NN, SVM, and random forest.

4.4. Discussion of the Results Obtained in Case A and Case B

In the previous sections we indicated that *ENN RENN* and *All-kNN* are representative of *Type-I* methods and for all of them we recorded very high levels of correlation between accuracy and compression. The minus sign of the correlation indicates that compression is inversely proportional to the accuracy, and this phenomenon was explained in the beginning of this chapter. The *RENN* and *All-kNN* are modifications of the basic *ENN* algorithm, so it is not surprising that all these methods behave similarly. The analysis of the code of *ENN* algorithm points out that its compression is equivalent to the leave-one-out estimation of the error rate of the *k*NN ($k = 3$) classifier, as all incorrectly classified samples by *k*NN ($k = 3$) are removed.

The *RENN* method repeats the *ENN* procedure until no instance is removed, so it often removes more samples than the basic *ENN* procedure. This imitates to some extent the *k*NN classifier with higher values of k (see Table 4), which explains why the correlation of *RENN* is higher than the one of *ENN*. The *All-kNN* rule also repeats the *ENN* step, but for a set of $k = [3 - 5]$ values (these settings were used in our calculations). Thus, it could not reflect as accurately the performance of the *k*NN classifier. It becomes especially noticeable when the best performance is obtained for large k ($k = 35$ or $k = 40$). Considering the computational complexity of these three algorithms where the *ENN* has the lowest ($O(n^2)$), followed by *All-kNN*, and the complexity of *RENN* reaches $O(n^3)$, the last two are definitely too high for the requirements of meta-learning systems. In this case we recommend the use of *ENN* instance selection as a landmark.

In case of condensation methods (*Type-II*) we analyzed the *CNN*, *IB2*, *RNGE*, *GGE* and *MSS* algorithms. The highest level of relation was obtained for *IB2* and *CNN*, which are very similar algorithms. In *CNN* the *IB2* procedure is repeated until all samples of the training set **T** are correctly classified by the *1*NN classifier trained on **P**. Hence, it uses the resubstitution error to determine **P**. In other words, both these methods try to maintain the performance of *1*NN classifier but we can not give any simple intuitive explanation as to why compression of these two methods is so significantly correlated to the prediction accuracy of the classifiers. The *GGE* and *RNGE* methods keep the border instances but using different criteria, as described in Section 2. The compression of these methods directly reflects the complexity of the decision boundary, or indicates how many instances have a neighbor from the opposite class. These types of measures were also discussed and analyzed by Cano in [53] who distinguished a special type in his analysis called *Fraction of Points on Class Boundary*, but unlike our analysis he used a minimum spanning tree to distinguish the fraction of border samples. He indicated that this type of measure has a *clear relation with classifier performance*. We also noticed this property but in our analysis the compression of *IB2* and *CNN* were much more informative than the number of border samples. With respect to *IB2* and *CNN*, the the former algorithm is more favorable, as it has higher correlation and lower computational complexity, which in the worst case can reach $O(n^2)$.

The *Drop* family as well as *ICF* did not reach such significant results. They are both designed to perform dataset cleansing as well as a condensation procedure. The compression does not reflect correctly the accuracy, because these two procedures are contradictory, as the cleansing methods have a high negative correlation while the condensation methods have a positive correlation. This is especially significant for the *ICF* algorithm, which directly applies the *ENN* algorithm (for which small compression corresponds to high accuracy) before condensation (high accuracy corresponds to high compression) which results in correlation almost equal to 0. In *Drop* the condensation dominates so it has positive correlation. The computational complexity of these two methods is also very high, reaching $O(n^3)$ so it also limits their possible applications to meta-learning.

5. Meta-Learning Applications Using Compression-Based Meta-Data

Previous experiments were focused on the analysis of individual relation between compression and accuracy for each instance selection method and classifier. The obtained results pointed out some limitations, which are related on one hand to a low correlation between compression and accuracy as in *ICF*, and on the other hand to computational complexity issues. Now, the question arises as to how compression-based meta-features work in real meta-learning problems. For this purpose we conducted another set of experiments as described below.

5.1. Experiment Setup

The experiments were designed to investigate the impact of the compression-based meta-features on the prediction quality of the meta-model. For that purpose four sets of meta-features were designed:

Set 1 State-of-the-art descriptors. This is the reference solution, which consists of 54 meta-features described in detail in Table 1.

Set 2 Compression-based descriptors. These include the compression of all instance selection methods, in total 13 meta-features.

Set 3 Compression-based descriptors. These include only the compression of the two methods suggested in the previous section, namely *ENN* and *IB2*.

Set 1+3 Combination of *Set 1* and *Set 3* descriptors. These are descriptors described in *Set 1* and *Set 3* to analyze the influence of compression-based meta-features on the state-of-the-art solution.

These sets on one hand provide a base rate, *Set 1*, and on the other hand analyze various scenarios of applications of compression-based meta-features.

The meta-model used in the experiments should take into account two properties:

- it should be an accurate nonlinear regression model, which can be applied to each of the metasets.
- it should return the feature importance indicator which would allow for assessment of the impact of each of the meta-features on the final results.

In order to satisfy the above properties we used *bagging of the regression trees* implemented in the Matlab Statistical and Machine Learning Toolbox. It is a very accurate regression model, which takes advantage of the regression trees, for which the feature importance index can be easily calculated. It is obtained by summing changes in the mean squared error resulting from the splits on every attribute, and dividing the sum by the number of branch nodes. The final index is aggregated over the ensemble.

The scheme of the experiments follows the one presented in Figure 2, except that it is embedded into the five times-repeated 10-fold cross-validation process. The detailed view of the test procedure is shown in Figure 6. The process starts by creating the metaset using 80 datasets described in Table 2. First for each of the datasets contained in the repository meat-features are extracted and every new meta-instance is labeled with the highest accuracy obtained for a given base-model (the so-called true performance). Then starts the five-times repeated cross-validation procedure which returns the root-mean-square error (RMSE). The error is calculated between the performance obtained for a given dataset contained in the repository estimated before the cross-validation procedure (the true performance) and the one returned by the meta-model. In the experiments we used the same set of the base-models which were used in the previous experiments, namely *k*NN, SVM, and random forest, with the same parameter settings (see Table 4). In total the process was executed 12 times (4 sets of descriptors $\times$ 3 types of base classifiers). Finally, the obtained results were evaluated using paired *t*-test to verify statistical significance of the difference between the results obtained from the reference *Set 1* and the remaining sets of descriptors containing compression-based meta-features. The default significance level ($\alpha = 0.05$) was used.

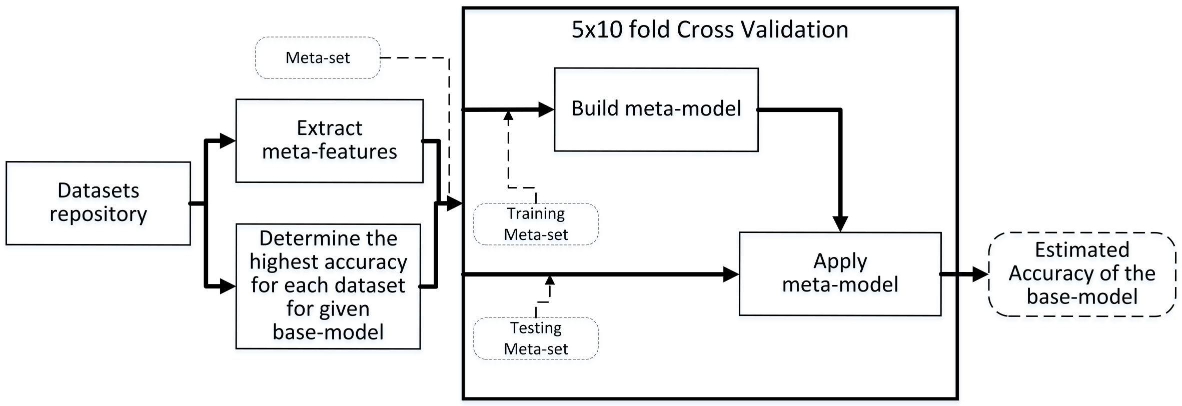

5x10 fold Cross Validation
Meta-set
Extract meta-features
Datasets repository
Determine the highest accuracy for each dataset for given base-model
Build meta-model
Training Meta-set
Apply meta-model
Testing Meta-set
Estimated Accuracy of the base-model

Figure 6. The scheme of the process used to compare the performances of the four metasets.

5.2. Results and Discussion

The obtained results presenting the minimum RMSE of the meta-model are shown in Table 7. The best results are marked in bold, and next to the results the *p*-value of the statistical test is reported.

Table 7. RMSE of accuracy estimation of the meta-model obtained for the four analyzed metasets and three classifiers *k*NN, SVM, and random forest. The best results are marked in bold.

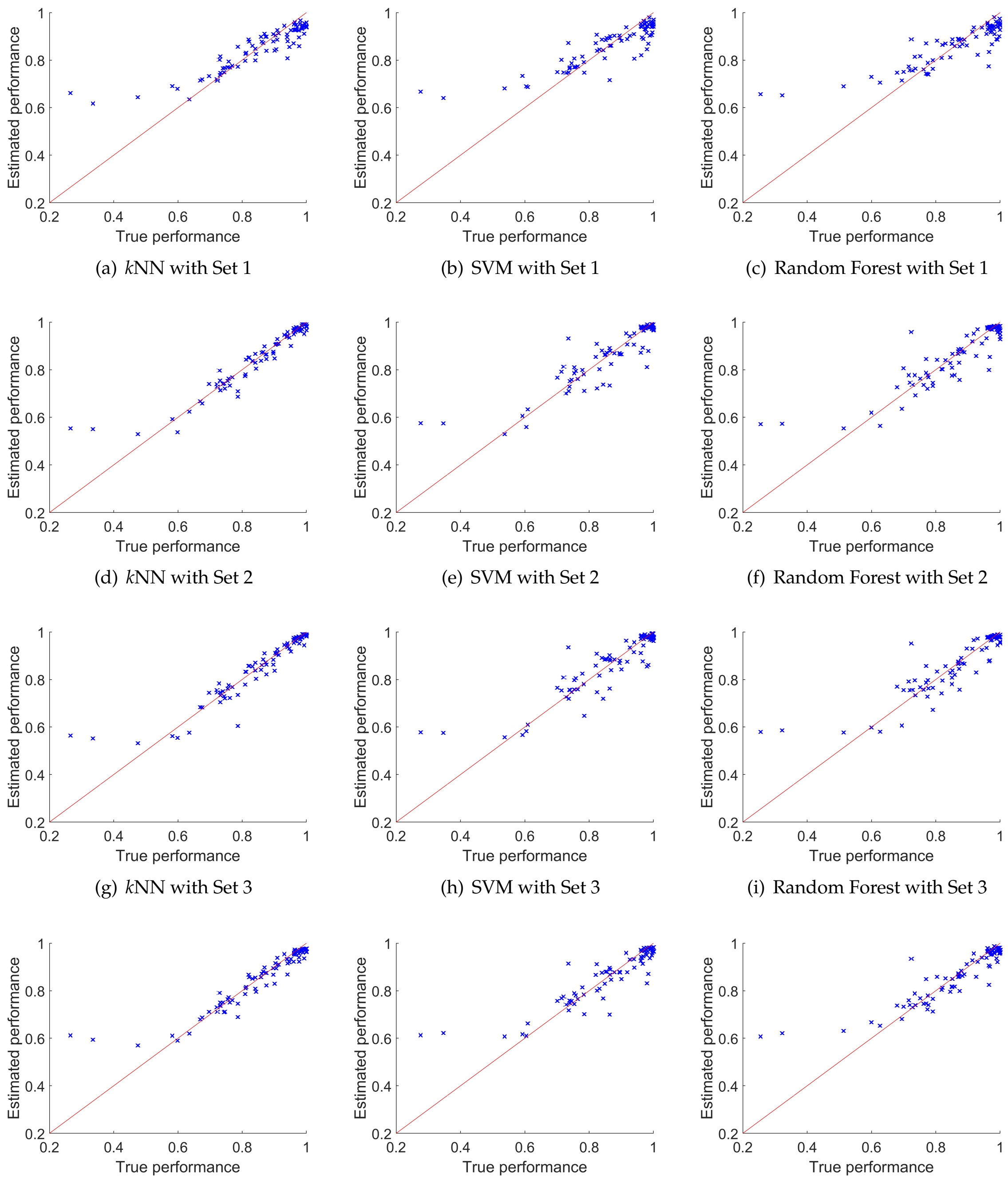
Meta Features	***k*NN**		**SVM**		**Random Forest**	
	RMSE ± std	***p*-Value**	**RMSE ± std**	***p*-Value**	**RMSE ± std**	***p*-Value**
Set 1	0.0671 ± 0.0343		0.0740 ± 0.0353		0.0760 ± 0.0382	
Set 2	**0.0394 ± 0.0288**	3×10^{-11}	**0.0580 ± 0.0302**	3×10^{-04}	0.0622 ± 0.0311	3×10^{-03}
Set 3	0.0431 ± 0.0305	5×10^{-05}	0.0586 ± 0.0294	1×10^{-03}	0.0628 ± 0.0292	4×10^{-03}
Set 1 + 3	0.0466 ± 0.0345	3×10^{-06}	0.0602 ± 0.0329	7×10^{-05}	**0.0604 ± 0.0351**	1×10^{-04}

The error rates indicate that the most accurate metasets according to the meta-model are those containing compression-based meta-features, which in all cases performed significantly better than the reference *Set 1*. Among metasets comprising compression-based descriptors, for the *k*NN classifier the most accurate system was obtained for *Set 2*. It is not surprising as the instance selection methods internally use the *k*NN classifier. The second most accurate system was obtained for *Set 3* and the third was obtained for *Set 1+3*. Similar results are obtained for SVM, here again the best set is *Set 2* followed by *Set 3* and *Set 1+3*, but the difference between *Set 2* and *Set 3* is very small. For random forest, the situation changes, and now the best is *Set 1+3*, in second place is *Set 2*, and *Set 3* takes third place.

The detailed view of the obtained results (from the five-times repeated cross-validation procedure) is also plotted in Figure 7. It shows 12 plots, where each represents performance estimated from the meta-model as a function of true performance for each of the 80 datasets denoted using the x mark (these values are averaged over five runs of cross-validation tests). The rows of Figure 7 represent results obtained for meta-model trained using different metasets. They are, respectively, row 1: *Set 1*; row 2: *Set 2*; row 3: *Set 3*; and row 4: *Set 1+3*. Columns represent different base classifiers, column 1: *k*NN; column 2: SVM; and column 3: random forest.

These results indicate that for all base classifiers every meta-model overestimates results for the datasets with poor accuracy (below 60%), but this effect is less apparent for datasets containing compression-based meta-descriptors where the estimated performances are closer to the red line. Similarly, for high performances (close to 1) for all base classifiers trained on *Set 1* (first row) the performance is underestimated—see the blue cloud of points close to $(1, 1)$ coordinates which are under the red line. This effect does not appear for the remaining rows, that is for metasets containing compression-based meta-features. The red line in the plots indicates perfect match between true and estimated performances.

To analyze which of the meta-features are the most significant for the meta-model we trained 12 meta-models (for each metaset and for each classifier) on the entire metaset (without splitting the training and test part), then we extracted the feature importance index. The indexes were then aggregated by averaging their values and plotted, as a result of which we obtained one figure per metaset (see Figure 8). The averaging process was required because for *Set 1* and *Set 1+3* we noticed small changes in order according to the classifier, such that, for example, for SVM the *dev. attr.* < *avg. skewness* and for *k*NN the opposite relation occurred with *dev. attr.* > *avg. skewness* without significant differences in the value of the indicators. For *Set 1* and *Set 1+3* only the top 22 features were shown in figures to make it readable.



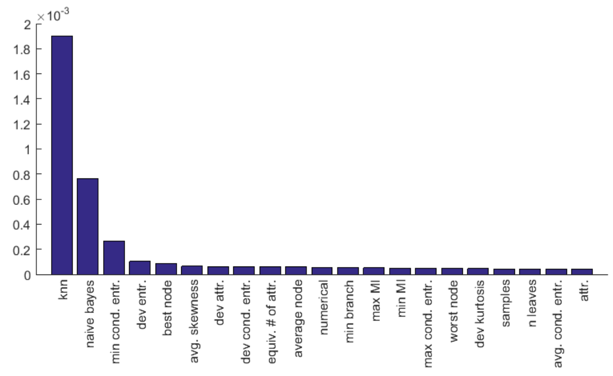
(a) *k*NN with Set 1 (b) SVM with Set 1 (c) Random Forest with Set 1

(d) *k*NN with Set 2 (e) SVM with Set 2 (f) Random Forest with Set 2

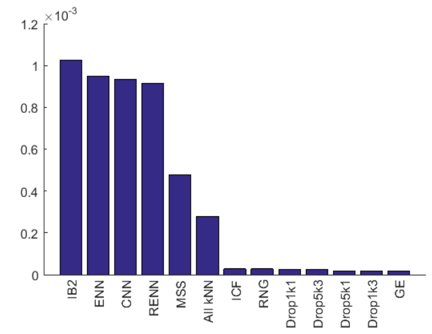
(g) *k*NN with Set 3 (h) SVM with Set 3 (i) Random Forest with Set 3

(j) *k*NN with Set 1+3 (k) SVM with Set 1+3 (l) Random Forest with Set 1+3

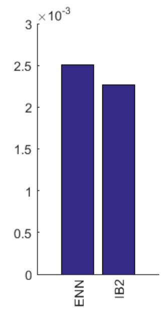
Figure 7. Comparison of true performance (X-axis) with performance estimated using a meta-model trained on three different metasets (Y-axis) for all base classifiers. Each x represents a single dataset, which is an average of the results obtained out of test sets of the cross-validation procedure (average over five repetitions).



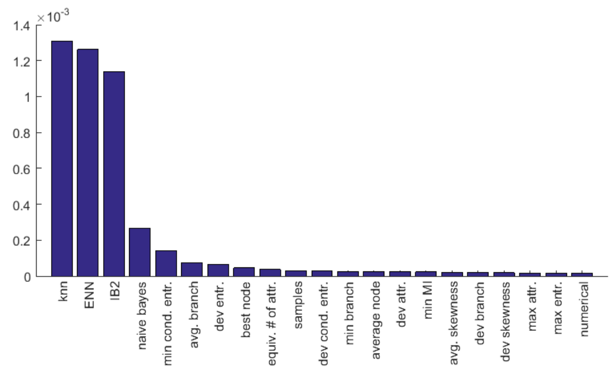
(a) Set 1: Reference descriptors



(b) Set 2: All compression-based descriptors



(c) Set 3: Selected compression-based descriptors



(d) Set 1+3: Combined Set 1 and Set 3

Figure 8. Meta-features importance indicators obtained from decision tree for each of the four analyzed sets of meta descriptors.

Meta-feature importance indicators gathered for *S*et 1 point out that the most valuable attribute is the performance of *k*NN, which is over two times more important than the *naive Bayes*-based landmark and over six times more important than the *minimal conditional entropy*. For *Set 2* the most important were compressions obtained by *IB2*, *ENN*, *CNN*, and *RENN* which had approximately equal indexes, followed by *MSS* and *All-kNN*. The remaining ones have indicators very close to 0. Note that *Set 2* was used just for comparison, and because of computational complexity it is not applicable ina real live scenario, despite the fact that it allowed the best results for *k*NN and SVM.

In case of *Set 3*, both meta-features were almost equally important. Finally, for the *Set 1+3* the indexes for *k*NN, *ENN*, and *IB2* were all the most important, followed by *naive Bayes*, which had a four times smaller index than any of the top three. The next one was *min. conditional entropy*, similarly to *Set 1*, with the rest being almost equal 0.

The obtained results show the validity of compression-based meta-features in application to meta-learning problems. In particular, the collections of meta-features described in *Set 3* and *Set 1+3* are important, as these two are applicable in real live scenarios. The two meta-features recommended in Section 4, which are based on compression of *ENN* and *IB2* work sufficiently well to accurately estimate the accuracy. The obtained error rates were in all cases at least 20% better than the results of the state-of-the-art solution with comparable computational complexity.

6. Conclusions

Model selection is a challenging problem in all machine learning tasks, and meta-learning has emerged as a tool to solve it. However, it requires good meta-data descriptors, which on one hand should have predictive power and on the other hand should be calculated very efficiently—much faster than the real prediction model. In this paper we investigated the use of compression of instance selection methods as a landmark for meta-learning systems. These type of meta-descriptors have a great advantage, because compression can be obtained for free as the instance selection is commonly used as a preprocessing step in many applications, and if not it can be calculated with at most $O(n^2)$.

The first part of the conducted experiments confirms that not all instance selection methods can be used as landmarks, and the relation between compression and accuracy depends on the type of instance selection. All methods denoted as *Type-I* are noise filters characterized with a negative correlation, while *Type-II*—the condensation methods—indicated positive correlation. Thus, the compression of the mixture of these two usually has a poor relation with accuracy, as illustrated by *ICF* algorithm. In this part of the research we distinguished the most promising instance selection algorithms, such as *CNN*, *IB2*, *ENN*, *RENN*, and *All-kNN*. Further analysis of these algorithms taking into account theoretical analysis of computational complexity indicated some limitations, which are reflected by a too-long execution time, limiting the most promising group just to two algorithms: the *IB2* and *ENN*.

The further research on real meta-learning applications indicates that compression of the proposed two instance selection methods (*IB2* and *ENN*) complement the state-of-the-art meta-data descriptors very well, and allow the user to achieve much lower error rates in estimating prediction accuracy of *k*NN, SVM, and random forest. These results were also confirmed by the analysis of the attribute importance index extracted from the meta-model which in all analyzed cases emphasized compression-based meta-features. This allows us to state that the compression-based meta-features should be used as a complement to traditional meta-learning systems, and confirms the thesis stated in the introduction.

The topic of the influence of the datasets already compressed by any of the instance selection methods on the values of compression-based meta-data descriptors was not covered in this research, but in the most of the cases it can be assumed that the data were not edited before meta-learning analysis. If this assumption is not valid, then it influences not only the compression-based descriptors but also all other meta-features, including information theory and other landmark methods. However, further analysis and research on this subject would be interesting.

Supplementary Materials: All of the datasets and RapidMiner scripts used to perform the experiments are available online at www.mdpi.com/1099-4300/19/11/583/s1.

Acknowledgments: I would like to acknowledge Mirek Kordos for his comments and help in the preparation of this manuscript.

Conflicts of Interest: The author declares no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	condensed nearest neighbor
IB2	instance-based learning type 2
GGE	Gabriel graph editting
*k*NN	*k*-nearest neighbors
SVM	support vector machine
ENN	edited nearest neighbor
RENN	repeated edited nearest neighbor
ICF	incremental case filtering
Drop	decremental reduction optimization procedure
RNGE	relative neighbor graph editing
MSS	modified selective subset

References

1. Abellán, J.; Castellano, J.G. Improving the Naive Bayes Classifier via a Quick Variable Selection Method Using Maximum of Entropy. *Entropy* **2017**, *19*, 247.
2. Grzymala-Busse, J.W.; Mroczek, T. A comparison of four approaches to discretization based on entropy. *Entropy* **2016**, *18*, 69.
3. Witten, I.H.; Frank, E.; Hall, M.A.; Pal, C.J. *Data Mining: Practical Machine Learning Tools and Techniques*; Morgan Kaufmann Publishers: San Francisco, CA, USA, 2016.
4. Zegers, P.; Frieden, B.R.; Alarcón, C.; Fuentes, A. Information Theoretical Measures for Achieving Robust Learning Machines. *Entropy* **2016**, *18*, 295.
5. Vilalta, R.; Giraud-Carrier, C.G.; Brazdil, P.; Soares, C. Using Meta-Learning to Support Data Mining. *IJCSA* **2004**, *1*, 31–45.
6. Castiello, C.; Castellano, G.; Fanelli, A. Meta-data: Characterization of Input Features for Meta-learning. In *Modeling Decisions for Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3558, pp. 457–468.
7. Fuernkranz, J.; Petrak, J. An Evaluation of Landmarking Variants. Available online: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.3221 (accessed on 27 October 2017).
8. Garcia, S.; Derrac, J.; Cano, J.R.; Herrera, F. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 417–435.
9. Blachnik, M. On the Relation Between kNN Accuracy and Dataset Compression Level. *LNAI* **2016**, *9692*, 541–551.
10. Haussler, D.; Kearns, M.; Schapire, R.E. Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension. *Mach. Learn.* **1994**, *14*, 83–113.
11. Hart, P. The condensed nearest neighbor rule. *IEEE Trans. Inf. Theory* **1968**, *16*, 515–516.
12. Aha, D.; Kibler, D.; Albert, M. Instance-Based Learning Algorithms. *Mach. Learn.* **1991**, *6*, 37–66.
13. Bhattacharya, B.; Poulsen, R.; Toussaint, G. Application of Proximity Graphs to Editing Nearest Neighbor Decision Rules. Available online: http://www-cgrl.cs.mcgill.ca/~godfried/teaching/mir-reading-assignments/Nearest-Neighbor-Editing.pdf (accessed on 27 October 2017).
14. Wilson, D. Assyptotic properties of nearest neighbour rules using edited data. *IEEE Trans. Syst Man Cybern.* **1972**, *SMC-2*, 408–421.
15. Tomek, I. An experiment with the edited nearest-neighbor rule. *IEEE Trans. Syst. Man Cybern.* **1976**, *6*, 448–452.
16. Wilson, D.; Martinez, T. Reduction techniques for instance-based learning algorithms. *Mach. Learn.* **2000**, *38*, 257–268.
17. Brighton, H.; Mellish, C. Advances in instance selection for instance-based learning algorithms. *Data Min. Knowl. Discov.* **2002**, *6*, 153–172.
18. Barandela, R.; Ferri, F.J.; Sánchez, J.S. Decision boundary preserving prototype selection for nearest neighbor classification. *Int. J. Pattern Recognit. Artif. Intell.* **2005**, *19*, 787–806.
19. Ritter, G.; Woodruff, H.; Lowry, S.; Isenhour, T. An algorithm for a selective nearest neighbor decision rule. *IEEE Trans. Inf. Theory* **1975**, *21*, 665–669.

20. Wang, X.Z.; Wu, B.; He, Y.L.; Pei, X.H. Nrmcs: Noise removing based on the mcs. In Proceedings of the 2008 International Conference on Machine Learning and Cybernetics, Kunming, China, 12–15 July 2008; pp. 89–93.
21. Marchiori, E. Class conditional nearest neighbor for large margin instance selection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 364–370.
22. Jankowski, N.; Grochowski, M. Comparison of Instance Selection Algorithms. I. Algorithms Survey. *Lect. Notes Comput. Sci.* **2004**, *3070*, 598–603.
23. Grochowski, M.; Jankowski, N. Comparison of Instance Selection Algorithms. II. Results and Comments. *LNCS* **2004**, *3070*, 580–585.
24. Olvera-López, J.A.; Carrasco-Ochoa, J.A.; Martínez-Trinidad, J.F.; Kittler, J. A review of instance selection methods. *Artif. Intell. Rev.* **2010**, *34*, 133–143.
25. Arnaiz-González, Á.; Díez-Pastor, J.; Rodríguez, J.J.; García-Osorio, C.I. Instance selection for regression: Adapting DROP. *Neurocomputing* **2016**, *201*, 66–81.
26. Arnaiz-González, Á.; Díez-Pastor, J.; Rodríguez, J.J.; García-Osorio, C.I. Instance selection for regression by discretization. *Expert Syst. Appl.* **2016**, *54*, 340–350.
27. Rodríguez-Fdez, I.; Mucientes, M.; Bugarín, A. An instance selection algorithm for regression and its application in variance reduction. In Proceedings of the 2013 IEEE International Conference on Fuzzy Systems (FUZZ), Hyderabad, India, 7–10 July 2013; doi:10.1109/FUZZ-IEEE.2013.6622486.
28. Kordos, M.; Blachnik, M.; Białka, S. Instance Selection in Logical Rule Extraction for Regression Problems. *LNAI* **2013**, *7895*, 167–175.
29. Kordos, M.; Blachnik, M. Instance Selection with Neural Networks for Regression Problems. *LNCS* **2012**, *7553*, 263–270.
30. Abdulali, A.; Hassan, W.; Jeon, S. Stimuli-Magnitude-Adaptive Sample Selection for Data-Driven Haptic Modeling. *Entropy* **2016**, *18*, 222.
31. Shaker, A.; Hüllermeier, E. IBLStreams: A system for instance-based classification and regression on data streams. *Evol. Syst.* **2012**, *3*, 235–249.
32. Czarnowski, I.; Jędrzejowicz, P. Ensemble online classifier based on the one-class base classifiers for mining data streams. *Cybern. Syst.* **2015**, *46*, 51–68.
33. Beringer, J.; Hüllermeier, E. Efficient instance-based learning on data streams. *Intell. Data Anal.* **2007**, *11*, 627–650.
34. Buza, K.; Nanopoulos, A.; Schmidt-Thieme, L. Insight: Efficient and effective instance selection for time-series classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 149–160.
35. Xi, X.; Keogh, E.; Shelton, C.; Wei, L.; Ratanamahatana, C.A. Fast time series classification using numerosity reduction. In Proceedings of the 23rd international conference on Machine learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 1033–1040.
36. Arnaiz-González, Á.; Blachnik, M.; Kordos, M.; García-Osorio, C. Fusion of Instance Selection Methods in Regression Tasks. *Inf. Fusion* **2016**, *30*, 69–79.
37. García-Osorio, C.; de Haro-García, A.; García-Pedrajas, N. Democratic instance selection: A linear complexity instance selection algorithm based on classifier ensemble concepts. *Artif. Intell.* **2010**, *174*, 410–441.
38. Blachnik, M. Ensembles of Instance Selection Methods Based on Feature Subset. *IEEE Proc. Comput. Sci.* **2014**, *35*, 388–396.
39. Garcia-Pedrajas, N. Constructing Ensembles of Classifiers by means of Weighted Instance Selection. *IEEE Trans. Neural Netw.* **2009**, *20*, 258–277.
40. Blachnik, M.; Kordos, M. Bagging of Instance Selection Algorithms. *LNAI* **2014**, *8468*, 40–51.
41. Leyva, E.; González, A.; Pérez, R. Knowledge-based instance selection: A compromise between efficiency and versatility. *Knowl. Based Syst.* **2013**, *47*, 65–76.
42. Leyva, E.; Caises, Y.; González, A.; Pérez, R. On the use of meta-learning for instance selection: An architecture and an experimental study. *Inf. Sci.* **2014**, *266*, 16–30.
43. Bensusan, H.; Kalousis, A. Estimating the Predictive Accuracy of a Classifier. In*Principles of Data Mining. Undergraduate Topics in Computer Science*; Springer: London, UK, 2013.
44. Brazdil, P.; Soares, C.; Pinto da Costa, J. Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results. *Mach. Learn.* **2003**, *50*, 251–277.

45. Thornton, C.; Hutter, F.; Hoos, H.H.; Leyton-Brown, K. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11–14 August 2013; pp. 847–855.
46. Mantovani, R.G.; Rossi, A.L.; Vanschoren, J.; Bischl, B.; Carvalho, A.C. To tune or not to tune: recommending when to adjust SVM hyper-parameters via meta-learning. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015; doi:10.1109/IJCNN.2015.7280644.
47. Rendell, L.; Seshu, R.; Tcheng, D.L. Concept-Learning and Dynamically-variable Bias Management. In Proceedings of the 10th International Joint Conference on Articial Intelligence, Milan, Italy, 23–29 August 1987; pp. 308–314.
48. Sohn, S. Meta Analysis of Classication Algorithms for Pattern Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **1999**, *21*, 1137–1144.
49. Gama, J.; Brazdil, P. Characterization of classification algorithms. *Prog. Artif. Intell.* **1995**, pp. 189–200.
50. Brazdil, P.; Gama, J.; Henery, B. Characterizing the applicability of classification algorithms using meta-level learning. In *European Conference on Machine Learning*; Springer: Berlin/Heidelberg, Germany, 1994; pp. 83–102.
51. Pfahringer, B.; Bensusan, H.; Giraud-Carrier, C. Meta-learning by Landmarking Various Learning Algorithms. In Proceedings of the 17th International Conference on Machine Learning, Stanford, CA, USA, 29 June–2 July 2000; pp. 743–750.
52. Bensusan, H.; Giraud-Carrier, C.G.; Kennedy, C.J. A Higher-order Approach to Meta-learning. Available online: http://www.cs.bris.ac.uk/publications/Papers/1000471.pdf (accessed on 27 October 2017).
53. Cano, J.R. Analysis of data complexity measures for classification. *Expert Syst. Appl.* **2013**, *40*, 4820–4831.
54. Pinto, F.; Soares, C.; Mendes-Moreira, J. Towards automatic generation of metafeatures. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*; Springer: Cham, Switzerland, 2016; pp. 215–226.
55. Reif, M.; Shafait, F.; Dengel, A. Meta-learning for evolutionary parameter optimization of classifiers. *Mach. Learn.* **2012**, *87*, 357–380.
56. Sun, Q.; Pfahringer, B. Pairwise meta-rules for better meta-learning-based algorithm ranking. *Mach. Learn.* **2013**, *93*, 141–161.
57. Grąbczewski, K.; Jankowski, N. Saving time and memory in computational intelligence system with machine unification and task spooling. *Knowl. Based Syst.* **2011**, *24*, 570–588.
58. Grabczewski, K.; Jankowski, N. Versatile and efficient meta-learning architecture: Knowledge representation and management in computational intelligence. In Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining, Honolulu, HI, USA, 1 March–5 April 2007; pp. 51–58.
59. Jankowski, N.; Duch, W.; Grąbczewski, K. *Meta-Learning in Computational Intelligence*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 358.
60. Alcalá, J.; Fernández, A.; Luengo, J.; Derrac, J.; García, S.; Sánchez, L.; Herrera, F. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Mult. Valued Logic Soft Comput.* **2010**, *17*, 255–287.
61. Asuncion, A.; Newman, D. UCI Machine Learning Repository. 2007. Available online: http://www.ics.uci.edu/~mlearn/MLRepository.html (accessed on 31 October 2017).
62. Stanfill, C.; Waltz, D. Toward memory-based reasoning. *Commun. ACM* **1986**, *29*, 1213–1228.
63. Wilson, D.R.; Martinez, T.R. Improved Heterogeneous Distance Functions. *arXiv* **1997**, arXiv:cs/9701101.
64. Grabczewski, K.; Jankowski, N. Transformations of symbolic data for continuous data oriented models. In *Artificial Neural Networks and Neural Information Processing—ICANN/ICONIP 2003*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 359–366.
65. Burges, C.J.C. Support Vector Machines and Kernel Based Methods. *Data Min. Knowl. Discov.* **1998**, *2*, 121–167.
66. Bottou, L.; Lin, C.J. Support vector machine solvers. In *Large Scale Kernel Machines* ; The MIT Press: Cambridge, MA, USA, 2007; pp. 301–320.
67. Blachnik, M.; Kordos, M. Simplnifying SVM with Weighted LVQ Algorithm. *LNCS* **2011**, *6936*, 212–219.
68. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32.
69. Grąbczewski, K.; Duch, W. Heterogeneous forests of decision trees. *LNCS* **2002**, *2415*, 504–509.

70. Kordos, M.; Blachnik, M.; Perzyk, M.; Kozłowski, J.; Bystrzycki, O.; Gródek, M.; Byrdziak, A.; Motyka, Z. A Hybrid System with Regression Trees in Steel-making Process. In *Hybrid Artificial Intelligent Systems*; Corchado, E., Kurzyński, M., Woźniak, M., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6678.
71. Rapid-I. RapidMiner. Available online: http://www.rapid-i.com (accessed on 26 October 2017).
72. Blachnik, M.; Kordos, M. Information Selection and Data Compression RapidMiner Library. In *Machine Intelligence and Big Data in Industry*; Springer: Cham, Switzerland, 2016; pp. 135–145.