An overview of Stacking and research opportunities about this meta-learner

HUGO PINTO*, Universidade Federal de Alfenas, Brazil
HUMBERTO BRANDÃO, Universidade Federal de Alfenas, Brazil

Researchers have been trying to improve the quality of machine learning estimators, combining them using several methods for decades. It is clear that combinations can improve results, but an important issue to be raisen is how to make it consistently and automate this process, without trying several possibilities for every single handled problem. In this context, the technique called Stacking has become more popular since this method is able to produce intelligent ensembles by itself, reducing considerably human efforts in the machine learning process. In this work, we present an overview of research works, which have applied Stacking, as well as new research possibilities, generating perspectives on this field.

Additional Key Words and Phrases: Stacked generalization, ensemble, classification, regression.

ACM Reference Format:
Hugo Pinto and Humberto Brandão. 2018. An overview of Stacking and research opportunities about this meta-learner. *ACM* 1, 1 (March 2018), 9 pages. https://doi.org/0000001.0000001_2

1 INTRODUCTION

A clear objective in the artificial intelligence area is to enable the reduction of time that people spend solving real problems [Sebastiani 2002]. In this context, there are many initiatives in completely different applications; *e.g.*, driving cars without human interaction, fraud detection, image recognition, music and movie recommendation and so on.

For decades, researchers have been trying to improve the quality of machine learning estimators, combining them in general frameworks, commonly called ensembles [Polikar 2006]. However, the process for creating good ensembles has been taking time and money from researchers and companies, simply because it is considered complex to mix estimators consistently. Taking time from humans is against the cited principle of the Artificial Intelligence, and, for this reason, a new field has emerged, being proposed by Wolpert [1992]. At first, Wolpert called his method as stacked generalization and, nowadays, it is also known as stacking. In a general explanation, running some estimators based on supervised learning, and applying a new layer of machine learning over the previous results, was the manner Wolpert proposed to transfer the hard work from humans to machines in order to combine different estimators.

Over the last two decades, machine learning has been faced with an increasing amount of supervised learning algorithms as well as the computing power [Lemke et al. 2015]. When Wolpert created stacking, the method was little used and spread because of some limitations. After recent advances, stacking has become more popular for two main reasons: (*i*) running more estimators and creating new learning layers take time, but new technologies have been considered important to handle it, and, (*ii*) the diversity generated by different models can improve the quality of ensembles [Oza and Tumer 2008; Polikar 2006].

2 STACKED GENERALIZATION

Trying to improve supervised learning processes, ensembles combine some predictions from machine learning algorithms, unifying them in a single answer. It is common to find ensembles aggregating outputs from base-estimators, generally using voting system, mean, median, among other manners [Jurek et al. 2014]. In this way, ensembles create outputs based on some estimators' predictions, often increasing the quality of the final answer. Generally they are able to improve accuracy and generalization power, when compared to the best base-estimator combined. These are some of the most important objectives of machine learning methods, such as presented in many works, *e.g.*, King et al. [2015], Anifowose et al. [2015], Hu and Tsoukalas [2003].

We can classify ensembles as homogeneous and heterogeneous. Homogeneous ensembles use the same base-estimator method in order to produce their results, *e.g.* bagging [Breiman 1996a] and boosting [Freund and Schapire 1995]. Among other ensemble methods, Stacking is considered a heterogeneous technique [Gupta and Thakkar 2014; Sesmero et al. 2015]. Heterogeneous ensembles use different learning algorithms for producing their final guesses [Ting and Witten 1997b]. Using more than one approach, heterogeneous algorithms can explore diversity more frequently if compared to a homogeneous one [Gupta and Thakkar 2014; Sesmero et al. 2015]. Besides that, stacking can also use Bagging and Boosting in its base-estimators, getting the advantages that homogeneous ensembles also provide.

Stacking approach divides the learning process in layers, which we could perceive that, until now, most of proposals have used a 2-layer algorithm. The first layer, called *level-0* in [Wolpert 1992], contains base-estimators, also known as models, such as linear regressions, decision trees, neural networks, naïve bayes, support vector machines, etc. The output of each instantiated model is used as an input for the next layer that is responsible for learning from these guesses, which were taken from different methods. In this way, the second layer, known as *level-1*, contains a machine learning method (meta-learner) and its responsibility is to create a smart ensemble, which learns to combine the outputs from the *level-0*, and which guesses should be used, according to the context of

*This is the corresponding author

Authors' addresses: Hugo Pinto, Universidade Federal de Alfenas, Alfenas, MG, Brazil, hugo.lcpinto@gmail.com; Humberto Brandão, Universidade Federal de Alfenas, Alfenas, MG, Brazil, humberto.brando@gmail.com.

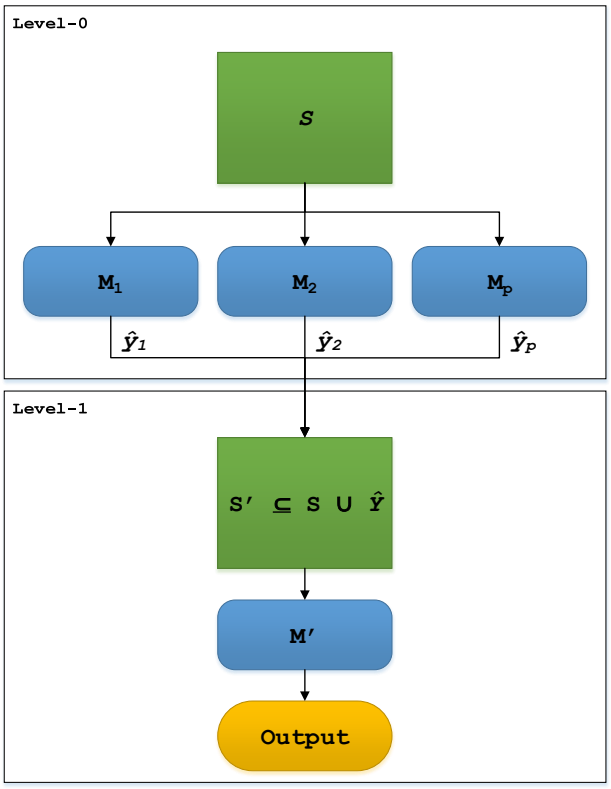

Fig. 1. A general framework flow for a 2-layer stacking.

	f1	f2	f3	f4	y
x1	3.4	2.8	1.0	3.6	1
x2	2.8	2.9	1.2	3.5	0
x3	4.9	1.9	1.0	2.8	1
x4	3.0	3.2	0.9	2.8	0
x5	4.8	2.9	0.8	3.0	0
x6	1.2	2.1	0.9	3.2	1
x7	0.4	1.0	1.1	2.9	0
x8	2.6	2.7	1.3	3.9	1
x9	1.9	3.0	1.2	3.1	0

Fig. 2. Example of an original space S.

	f1	f2	f3	f4	ŷ1	ŷ2	y
x1	3.4	2.8	1.0	3.6	0.9	0.8	1
x2	2.8	2.9	1.2	3.5	0.2	0.1	0
x3	4.9	1.9	1.0	2.8	0.6	0.5	1
x4	3.0	3.2	0.9	2.8	0.3	0.2	0
x5	4.8	2.9	0.8	3.0	0.3	0.0	0
x6	1.2	2.1	0.9	3.2	0.8	0.4	1
x7	0.4	1.0	1.1	2.9	0.6	0.5	0
x8	2.6	2.7	1.3	3.9	0.5	0.2	1
x9	1.9	3.0	1.2	3.1	0.0	0.1	0

Fig. 3. Example of an extended space.

each instance. A general flow of stacked generalization is shown in Figure 1.

2.1 Definitions

The space S, presented in Figure 1, and defined in Equation 1, contains instances for supervised learning algorithms, which are used at first in the *level-0*. Wolpert [1992] emphasizes that S contains the original search space. In this work, we define the number of instances as m and the number of original features as n. Each instance x_i for all i in $\{1, \ldots, m\}$ is a vector with n features, where each x_i is labelled by a scalar value y_i for all i in $\{1, \ldots, m\}$. As well as in Wolpert [1992], we define the learning problem only using a single output for instance, though it can be extended to support multiple outputs.

$$S = \cup_{i=1}^{m}(x_i, y_i) \tag{1}$$

Considering $m = 9$, $n = 4$ and the problem as a binary classification, we present an illustrative example of a *level-0* dataset S in Figure 2.

2.1.1 Level-0. In this work, we define the number of estimators used in the *level-0* as p. Each estimator M_j for all j in $\{1, \ldots, p\}$ will create m guesses represented by the vectors $\hat{y}_j$ for all j in $\{1, \ldots, p\}$. Creating p guesses for each instance in S, after running different instantiated models, the Stacking technique can provide diversity for taking better guesses in the *level-1*'s meta-learner. In Figure 3, we extended the illustrative example from Figure 2, considering $p = 2$.

In Figure 3, after running hypothetical two models, M_1 and M_2, over the space S, we can see two new columns, $\hat{y}_1$ and $\hat{y}_2$, which represent guesses from M_1 and M_2 respectively. These artificial features, $\hat{y}_1$ and $\hat{y}_2$, are also known as meta-features.

Once the first phase is done, the p vectors of guesses generated by base-estimators are used to compose the *level-1* dataset (meta-dataset S'). Each vector $\hat{y}_j$ in $j = \{1, \ldots, p\}$ is a meta-feature in the new and transformed space S', where each one contains m guesses. In this point, we have two possibilities: either S' includes the original features or not. Section 3.1.3 presents works that use the original space of features, in which the meta-learner is able to analyses it context using it. Section 5.5 describes advantages, disadvantages and research opportunities also related to this specific issue.

In this work, the most common manners that estimators create these m outputs for each instantiated model are presented in Section 2.1.3. This is a very important part of Stacking in order to avoid overfitting. A discussion of possibilities to generate these outputs as current research opportunities are presented in Section 5.4.

2.1.2 Level-1. After constructing a meta-dataset S', the *level-1* meta-learner M' should be generated using any learning algorithm

[Sesmero et al. 2015]. *Level-1* contains at least one meta-learner that uses, as inputs, the outputs generated by *level-0*.

It is common to see cross-validation approaches being applied during *level-1* phase in order to identify good parameters to be used in the meta-learner M'. If the stacking is formed exactly from two layers, the final guesses are taken from the meta-learner M'. Some works have reported the use of more than two layers and we discuss them in the Section 3.3.

2.1.3 Generating guesses during level-0. During *level-0*, in order to avoid overfitting, the *k-fold* method and its variations are the most used techniques employed to generate guesses during *level-0*. Splitting the dataset into two folds, one for training and a hold-out for testing, can also be used at this phase of Stacking. These strategies are mainly used to obtain *out-of-fold* predictions that will be used at the meta-level [Arlot et al. 2010].

K-fold method. The *k-fold* method has been widely applied by several authors [Breiman 1996b; Menahem et al. 2009; Ting and Witten 1997a]. The usage of this method enables the meta-dataset to have the same number of samples of the *level-0* dataset S. During this step, the algorithm divides the set S into k roughly equal parts $(P_1, P_2, \ldots P_k)$. The vector of m guesses of each estimator i ($\hat{y}_i$), for all i in $\{1, \ldots, p\}$, is generated using k training processes. Let us suppose $k = 3$ and $i = 1$, then S is divided into three subsets, P_1, P_2 and P_3, in order to generate three out-of-fold predictions. As shown in Figure 4, the first out-of-fold of $\hat{y}_1$ is created using a supervised learning process of the estimator number one using the $P_2 \cup P_3$ as its training set.

	f_1	f_2	f_3	f_4	$\hat{y}_1$	$\hat{y}_2$	y
x_1	3.4	2.8	1.0	3.6	0.9		1
x_2	2.8	2.9	1.2	3.5	0.2		0
x_3	4.9	1.9	1.0	2.8	0.6		1
x_4	3.0	3.2	0.9	2.8			0
x_5	4.8	2.9	0.8	3.0			0
x_6	1.2	2.1	0.9	3.2			1
x_7	0.4	1.0	1.1	2.9			0
x_8	2.6	2.7	1.3	3.9			1
x_9	1.9	3.0	1.2	3.1			0

Fig. 4. First out-of-fold generated by M_1 using K-fold.

Using the same parameters and the same algorithm, the second out-of-fold of $\hat{y}_1$, is generated, but, this time, using $P_1 \cup P_3$ as its training set (Figure 5).

In this example, the last out-of-fold needs to be generated using $P_1 \cup P_2$ as its training set, as shown in Figure 6.

This process is necessary for all p estimators used in the *level-0*, generating, at the final, a complete table as shown in Figure 3. It avoids the inclusion of any output information in meta-features. In several works related to machine learning, we can see *k-fold* method used as a validation process in order to set and choose estimators. Despite some works have used the term *k-fold cross-validation* to define this column generation procedure in stacking, it is not a validation process. However, some works have also reported the use

	f_1	f_2	f_3	f_4	$\hat{y}_1$	$\hat{y}_2$	y
x_1	3.4	2.8	1.0	3.6	0.9		1
x_2	2.8	2.9	1.2	3.5	0.2		0
x_3	4.9	1.9	1.0	2.8	0.6		1
x_4	3.0	3.2	0.9	2.8	0.3		0
x_5	4.8	2.9	0.8	3.0	0.3		0
x_6	1.2	2.1	0.9	3.2	0.8		1
x_7	0.4	1.0	1.1	2.9			0
x_8	2.6	2.7	1.3	3.9			1
x_9	1.9	3.0	1.2	3.1			0

Fig. 5. Second out-of-fold generated by M_1 using K-fold.

	f_1	f_2	f_3	f_4	$\hat{y}_1$	$\hat{y}_2$	y
x_1	3.4	2.8	1.0	3.6	0.9		1
x_2	2.8	2.9	1.2	3.5	0.2		0
x_3	4.9	1.9	1.0	2.8	0.6		1
x_4	3.0	3.2	0.9	2.8	0.3		0
x_5	4.8	2.9	0.8	3.0	0.3		0
x_6	1.2	2.1	0.9	3.2	0.8		1
x_7	0.4	1.0	1.1	2.9	0.6		0
x_8	2.6	2.7	1.3	3.9	0.5		1
x_9	1.9	3.0	1.2	3.1	0.0		0

Fig. 6. Third out-of-fold generated by M_1 using K-fold.

of the cross-validation metric to define if the column will be used in S' or not, similar to a feature selection method.

Stratified k-fold. Authors have also used *k-fold* variations, such as *stratified k-fold*. This method has the same main principals of standard *k-fold*, but when splitting the data into k roughly equal parts, each P_i, for i in $\{1, \ldots, k\}$, has to keep the same class distribution proportion as the original dataset S [Gama 1998; Gomes et al. 2012].

Leave-one-out. The *leave-one-out* method has also been used in Stacking literature [Ozay and Yarman-Vural 2016; Wolpert 1992]. In a simplistic explanation, *leave-one-out* can be seen as a *k-fold* with k been the same as the number of m examples in the dataset S. This technique used for generating guesses for the meta-level has been used with less frequency than the previous described in this work. This fact could be justified because of its high computational cost, as it generates m models for each p estimator.

Hold-out strategies. This strategy consists of splitting the dataset S into two folds, a hold-out and a fold for training. The first fold will compose the meta-level and will not be part of base-estimators' training. Hold-out strategies are generally employed to large datasets, in which removing a portion of it, as a hold-out, may not infer in a huge accuracy loss. Although, a precision degradation is expected [Sakkis et al. 2001] and, because of this, this method is less used than others, even though it spends less computing time than *k-folds* [Witten et al. 2011].

3 AN OVERVIEW OF STACKING

Since Wolpert [1992] first introduced stacked generalization for classification tasks, and Breiman [1996b] successfully applied it to the regression domain, several different approaches of how to adapt, extend or apply stacking were already proposed.

In this work, we present an overview of research works, which have used stacking, separating them into categories, based on our vision about how to customize the method. The diversity of possibilities to apply stacking exist because there is not a perfect manner of doing it.

Considering complexity for codifying, Bagging [Breiman 1996a] and Boosting [Freund and Schapire 1995], which are homogeneous ensembles methods, are simpler and are yet more widely used than stacking [Witten et al. 2011]. However, stacking can include bagging and boosting as base-estimators, besides giving the opportunity to apply machine learning in more layers. Using layers, stacking is also categorized as a meta-learning method [Jurek et al. 2014]. In a meta-learner, inputs are outputs from previous layers, which can improve the final results considerably [Lemke et al. 2015; Vilalta and Drissi 2002]. Mixing outputs from different models, meta-learners have potential to create good estimators, because the ensemble that combines outputs is generated using a machine learning process, instead of generating final guesses using average, median, etc. Of course, for running different algorithms as base-estimators, stacking requires computer power and the trade-off, between quality and time, needs to be considered.

3.1 Meta-level dataset construction

Even though there are several methods that can be used to build the level-1 dataset structure, the type of outputs from the base-estimators should also be considered. Is this section, we further detail some different approaches that have already been used since Wolpert proposed stacking.

Basically, there are two possible outputs from estimators, the *class categorical predictions* and the *class probabilities distribution*. However, other features can be extracted from base-estimators' out-of-fold predictions, in order to compose the *level-1* dataset. Such features can be the *entropy of the class probabilities*, as used by Džeroski and Ženko [2004]; Todorovski and Džeroski [2000, 2003]; Ženko et al. [2001], or *logarithmic conditional probabilities*, proposed by Bao et al. [1998].

3.1.1 Class categorical predictions. The first approach, developed by Wolpert [1992], uses crisp outputs from base-estimators. In this way, the *level-1* dataset will only contains discrete predictions as meta-features along with the correct label vector. Figure 7 presents a *level-1* dataset using the class categorical predictions structure, considering the outputs of 5 *level-0* estimators (*i.e.,* $p = 5$) and the correct classification y, for a binary problem.

The usage of categorical predictions allows the measurement of diversity and collaboration among base-estimators using some methods. Such measures directly imply on meta-learner's accuracy and are useful to understand the model as well [Fan et al. 1999]. An example of measure is the *shareability*, proposed by Ozay and Yarman-Vural [2016]. It evaluates the proportion of samples of the dataset that were correctly predict by, at least, one base-estimator. Other

	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_3$	$\hat{y}_4$	$\hat{y}_5$	y
x_1	1	1	1	1	1	1
x_2	0	0	1	0	0	0
x_3	1	1	1	1	1	1
x_4	0	0	0	0	0	0
x_5	0	0	1	0	0	0
x_6	1	0	0	0	1	1
x_7	1	1	1	1	1	0
x_8	1	0	0	0	1	1
x_9	0	0	0	0	0	0

Fig. 7. Class categorical predictions along with the correct label.

measurements were applied by Fan et al. [1999] to infer Stacking's overall accuracy. They have introduced the *conflict-based* measure, which evaluates the proportion of conflicts between base-estimators' outputs.

Besides the benefits of having ways to measure the meta-learner's performance, an issue could arise when using class categorical predictions. This issue comes up when a particular estimator does not automatically generate discrete predictions, but probabilities from 0.0 to 1.0. It is related to the selection of a threshold that will define if a given predicted probability should be used as a 0 or a 1. For each dataset, the threshold that will best distinguish one class from another may vary, mainly because of each dataset classes proportion. In real-world problems, new datasets class proportion are rarely known, making the threshold specification a complicated task. For instance, Figure 7 is resultant of Figure 8 applying a threshold of 0.5 in each probability.

3.1.2 Class probabilities distribution. Rather than using crisp predictions, one can employ the class probabilities distribution extracted from each base-estimator as meta-features. In this approach, the *level-0* models need to generate a probability prediction for each instance, which can vary from 0.0 to 1.0. For multi-class datasets, the *level-0* models can generate C probability predictions for each instance, where C is the number of classes. Figure 8 presents the *level-1* dataset structure using class probabilities distribution, considering the outputs of 5 *level-0* estimators (*i.e.,* $p = 5$) along with the correct classification y, for a binary problem.

Using not only discrete predictions from base-estimators, but the class probabilities distribution, allows the meta-estimator to take advantage of each *level-0* estimator's confidence, leading to more accurate predictions on *level-1*, as reported by Ting and Witten [1997a, 1999] and confirmed by many authors [Chen et al. 2009; Džeroski and Ženko 2002; Gama 1998; Menahem et al. 2009; Seewald 2002].

Even having the confidence of each base-estimator, the usage of class probabilities distribution, as *level-1* meta-features, can derive into huge datasets at *level-1* when the data is multi-class. The *level-1* dataset may contain $(C \times p)$ meta-features. In the other hand, the previous mentioned approach would generate just p meta-features. Such huge datasets were found, by Seewald [2002], to be a stacking's

	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_3$	$\hat{y}_4$	$\hat{y}_5$	y
x_1	0.9	0.8	0.8	0.5	0.9	1
x_2	0.2	0.1	0.5	0.0	0.4	0
x_3	0.6	0.5	0.9	0.9	0.9	1
x_4	0.3	0.2	0.1	0.2	0.0	0
x_5	0.3	0.0	0.6	0.0	0.3	0
x_6	0.8	0.4	0.4	0.1	0.9	1
x_7	0.6	0.5	0.4	0.8	0.9	0
x_8	0.5	0.2	0.1	0.1	0.9	1
x_9	0.0	0.1	0.0	0.2	0.2	0

Fig. 8. Class probabilities distribution along with the correct label.

weakness. In his experiments, stacking performed worst on multi-class datasets than binary ones. Menahem et al. [2009] have called it as *curse of dimensionality* in their work, also known as the Hughes effect [Hughes 1968].

3.1.3 Including original features. First employed by Chan and Stolfo [1993], this approach seeks to include more information to the *level-1* dataset, *i.e.*, the *level-0* dataset. This approach suggests that only meta-features obtained from *level-0* estimators are not sufficient to achieve the higher predictive accuracy and generalization by themselves. So, the meta-estimator can be benefited and improve results when also uses original features to make predictions.

This *level-1* dataset architecture can be seen as an extension of one of both previous strategies described. It can be built using class categorical predictions or class probabilities distribution along with the *level-0* dataset. Figure 3 shows this approach when employing class probabilities distribution along with *level-0* features.

Research on the stacking literature also reports that using *level-0* dataset along with the meta-features as *level-1* data may yield accuracy improvement [Ekbal and Saha 2013; Todorovski and Džeroski 2000; Torres-Sospedra et al. 2006; Vilalta and Drissi 2002]. Even that this approach enables the *level-1* estimator to have extra information to learn on, it has not been widely used and this may due to avoid the usage of huge datasets when generating the meta-learner. We discuss more about this approach, considering its advantages and disadvantages in the Section 5.5.

3.2 Setting stacking up using optimization methods

Even that Wolpert [1992] stated the need of knowledge about the learning algorithms to be used when configuring stacking, some approaches employ automatic methods to configure it. These methods try to select *level-0* and *level-1* estimators and their parameters. The search-based stacking approaches are mainly built using heuristic optimization methods to find good configurations, such as Genetic Algorithms (GA), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC) and so on.

A genetic-search approach was proposed by Ledezma et al. [2010] to be used with stacking, seeking for good configurations, called GA-Stacking. As the method adapts to the domain characteristics, the parameters found by it are domain-dependent, making the method flexible. Besides benefits, GA-Stacking takes more computing time to run in comparison to standard stacking, as reported by Ledezma et al. [2002]. Moudrík and Neruda [2015] also claimed that the main drawback of using genetic-search in a learning context is the time it consumes.

GA was used by [Ekbal and Saha 2013] to find the most relevant and promising features in their approach. Support Vector Machines and Conditional Random Field were the framework's base-estimators and were fed with the features found by GA. Different models using the feature selection scheme were generate in the first stage of their approach and then, in the second stage, combined using Stacked Generalization.

Chen et al. [2014] proposed another genetic-search base stacking approach, also domain-dependent. Their work employed the ACO meta-heuristic to find stacking good configurations. Experiments were done fixing C4.5 decision tree as *level-1* model and using ACO to find *level-0* configurations. And experiments using ACO to find both *level-0* and *level-1* configurations were done as well.

Shunmugapriya and Kanmani [2013] employed ABC heuristic in their research to find good stacking configurations. They adopted two different approaches, (*i*) fixing the *level-1* estimator as a J48 decision tree and search for *level-0* configurations with ACO and (*ii*) using ACO for both *level-0* and *level-1* estimators.

A comparison of different heuristic-based stacking was done by Gupta and Thakkar [2014] in order to analyze how to find appropriate *level-0* and *level-1* configuration. Approaches using evolutionary algorithms such as GA, ACO and ABC were analyzed, but other methods can yet be employed and studied to optimize stacking configurations, including Firefly and Particle Swarm Optimization.

3.3 Using more than 2 layers

The first stacking framework, describe by Wolpert [1992], has two levels, *level-0* and *level-1*. Some stacking approaches using more than only two layers have already been induced. Such approaches seek to relax the lower level estimators' bias at each level, in order to achieve better predictive accuracy.

Gama and Brazdil [2000] proposed a scheme called Cascade Generalization, which works sequentially. In this approach, the class probabilities distribution generated by each estimator is concatenated to the dataset, augmenting it iteratively. They research seeks to insert more information in the dataset for each model generated. When applying Cascade Generalization, several layers can be built using different learning algorithms.

Referred as Troika, Menahem et al. [2009] proposed a new stacking approach, which address multi-class problems in a four-layers stacking scheme. To begin with, the *level-0* classifiers provide class probability distribution predictions to fill the next level dataset. The first meta-layer, *level-1*, is composed of specialist classifiers, which are trained using one-against-one (OAO) binarization method. This method is used to avoid the curse of dimensionality. *Level-2* is composed of C classifiers, one for each class of the problem. They receive out-of-fold predictions of *level-1* classifiers as input. These classifiers are trained using one-against-all (OAA) method. The third, and last,

meta-layer (*level-3*) consists in just one classifier that yields out-of-fold predictions from every *level-2* classifier, and then, performs the final prediction.

4 DIVERSITY AND ITS IMPACTS

Ensemble methods are well-known to produce better overall accuracy when their base-estimators perform well, but are diverse at the same time [Oza and Tumer 2008]. Diverse estimators make errors at distinct instances' predictions, canceling out the combined outputs' mistakes [Polikar 2006]. Stacking is an ensemble method that allows the combination of estimators generated using different learning algorithms and, in this way, it can explore the diversity among *level-0* base-estimators.

According to [Dietterich 1997], there are four general methods that can be applied to any learning algorithm for constructing ensembles. The first method is to subsample the dataset, in order to generate multiple outputs, by running the same learning algorithm several times, varying the subset used for training. Both bagging [Breiman 1996a] and boosting [Freund and Schapire 1995] are included in this method, the first samples the train instances with replacement, while the second tries to correct predictions of misclassified examples at each iteration. Manipulate input features is the second general method, where it seeks to select a subset of the data attributes for each algorithm to learn on, in order to the estimators learn in distinct ways. The third general technique is to manipulate the predictions $\hat{y}_i$ of each base-estimator. This method is generally used to solve multi-class problems, by transforming them into multiple two-class (binary) tasks. Inject randomness into the learning algorithm is the last general method for constructing diverse ensembles reported by Dietterich [1997]. It can be achieved varying the inputs parameters of learning algorithms, e.g., initial weights for training neural networks [Ghorbani and Owrangh 2001; Kolen and Pollack 1991].

Besides general methods mentioned above, stacking allows the use of different learning algorithms to generate *level-0* estimators, in order to obtain diversity. The heterogeneous ensemble technique [Wolpert 1992] exploit the different biases about *level-0*, learning their characteristics when making predictions at the *level-1*. Many approaches seeking for achieving diversity using stacking has been proposed, as we present in the remaining of this section.

Genetic Algorithm (GA) was used to search for good stacking configurations by Ledezma et al. [2010]. In their work, a GA selected which and how many level-0 estimators should be used, as well as which *level-1* estimator and its parameters. In this domain-dependent approach, since the framework adapts to the domain in study, the number of *level-0* influenced the overall accuracy. The authors show that configurations with better overall accuracy were achieved using 9 to 10 *level-0* estimators, where 10 was the maximum possible amount.

Zhu [2010] proposed a hybrid approach to construct ensembles, integrating data envelopment analysis (DEA) and stacking. DEA was employed to perform base-model selection, while stacking had the combining task. The author claims that the approach success was achieved possibly due to an improved DEA model on the selection of diverse base-estimators.

In a domain-specific approach, Chen et al. [2014] proposed a method for ensemble construction, which employs an Ant Colony Optimization (ACO) to select stacking estimators. The authors performed experiments with and without the introduction of correlative differences among base-estimators before the ACO starts. The most promising approach was found to be the one using these correlative differences, as local information, in order to bring a more diverse combination of estimators at the base-layer.

Anifowose et al. [2015] proposed an Stacked Generalization approach composed of Support Vector Machines, where both *level-0* and *level-1* were SVM estimators. In order to achieve high diversity among the base-models, generated using the same learning algorithm, a different regularization parameter value was used for each estimator, as the technique is very sensitive to it.

In order to introduce diversity to stacking base-estimators, Ozay and Yarman-Vural [2016] generated nearest neighbor estimators in different feature spaces. Seeking to gain expertise on several characteristics of instances, different attributes were extracted from the data. Experiments with one-class expert estimators were done as well, where each estimator was expert on a specific class of the problem.

4.1 Applications

Stacked Generalization has been used to solve a wide range of real-world problems. In this section, we present some examples of these applications using stacking.

Tzanis et al. [2012] proposed StackTIS, a stacking framework used in the detection of translation initiation sites (TISs). In order to predict TIS, an open research problem in bioinformatics, the authors generated three *level-0* estimators. They entitled them as a coding component, a consensus component, and an upstream length component. The coding component was induced using SVM classifier. A 1st order homogeneous Markov chain was used as consensus component and the upstream length component has the task of calculate the distance of a potential TIS from the 5' end of a cDNA sequence. MLR and M5' were used at the *level-1* to combine the outputs of the three base-estimators, where M5' outperformed MLR. The authors claim that StackTIS performs significantly better than other TISs detection approaches in the literature, even of their previous study [Tzanis et al. 2007].

Ness et al. [2009] proposed an improvement to multiple label class classification for automatic music tag annotation using a framework with two levels. They employed SMVs at both *level-0* and *level-1* in their stacking approach. In order to benefit from the confidence of *level-0* estimators at *level-1*, class probabilities distribution was chosen for the meta-features structure. In the authors' experiments, their proposed method improved the performance of state-of-art methods in this field in the two evaluated datasets.

Stacking was employed by Sill et al. [2009] in Netflix Prize Competition, a product recommendation task, achieving the second-place. As well as in other works [Koren 2009; Töscher et al. 2009], the authors refer to stacking as a blender. They proposed the Feature-Weighted Linear Stacking (FWLS) approach, which combines linearly meta-features with original features, generating their vectors

$\hat{y}_j$ in $j = \{1, \ldots, p\}$. In other words, they create artificial columns that are not used directly in the space S'.

King et al. [2015] verified the predictive performance of several ensemble-learning methods for pay-per-click campaign management, which includes bagging, voting, meta-cost and stacking. They seek the combination of classification algorithms, such as naïve bayes, logistic regressions, decision trees and support vector machines (SVM). Stacking performed better than they expected, where they classified as excellent. According to them, in text mining, other researchers do not know stacking for getting good performances, then, in our opinion, it can be a research opportunity in this area.

In a petroleum reservoir characterization task, Anifowose et al. [2015] proposed a Stacked Generalization approach built using ten different support vector machines as base-estimators in *level-0* ($p = 10$) and another one as the *level-1* model. Varying SVM's input parameters was the manner employed by the authors in order to achieve diversity among them. In their opinion, the ensemble proposed by them, showed great potential and outperformed the other techniques in most of experiments. In order to compare, they used the following methods: random forest, SVM with the bagging method, and a standard SVM.

Over the Chinese board game called Go, Moudrík and Neruda [2015] employed stacking for predicting player's attributes, specifically to identify proficiency and style. Their non-linear stacking approach was compared with other methods, which were, mean regression, random forest, partial least squares, bagged neural network and hand-tuned learner. In order to set stacking up, they used the genetic algorithm for choosing base-classifiers and the meta-learner. The genetic search required a large amount of time according to them and, for this reason, they limited the experiments to contain, at most, five *level-0* base-estimators ($p \leq 5$). Compared with other used methods, the proposed stacking got the better results.

Noçairi et al. [2016] employed stacking in order to combine classifiers in a binary classification problem. Their approach was applied in the cosmetic industry domain, where the dangerousness of 165 chemicals were predicted. The *level-0* base-classifiers used in the study were partial least squares discriminant analysis, boosting, support vector machines, naïve bayes and expert scoring. Their stacking method has performed better than each *level-0* base-classifier.

Doumpos and Zopounidis [2007] employed Stacked Generalization to combine models in credit risk assessment. Their approach seeks to distinguish potential defaulters from non-defaulters. The predictions of seven *level-0* models were combined, which were generated using different methods, such as linear discriminant analysis, quadratic discriminant analysis, logistic regression, probabilistic neural networks, nearest neighbors, classification and regression trees and support vector machines. In order to decide which method should be used in the *level-1*, the authors tried the same that were used in the *level-0*. Besides that, instead of using directly all outputs from *level-0*, they applied the principal component analysis (PCA) method in order to use this new space of features as input for the *level-1*. Using stacking, this transformation was also tested by Merz [1999].

Stacking was employed by Xing et al. [2016] to predict dropouts in MOOCs (massive open online courses). In this field, the dataset is highly imbalanced, since more than 90% of the students tend to drop out the course, according to the authors. General bayesian network (GBN) and decision tree C4.5 were used as *level-0* classifiers of their stacking framework. In order to help instructors to provide targeted support to potential dropouts, predictions were made for each week of the course. Current week's data and appended historical data of previous weeks were used as the training dataset. In their work, stacked generalization outperformed GBN and C4.5 solving the problem.

Artificial neural networks (ANNs) were used by Hu and Tsoukalas [2003] trying to explain consumer behavior, identifying the relative importance of situational and demographic factors on this task. In order to create the *level-0*, 29 ANNs were generated using different configurations, which concerns in parameters of the algorithm and input variables. Each *level-1* model's training set was fed with five selected base-estimators outputs, generating 24 *level-1* models. The authors did not use all 29 outputs as inputs for the *level-1* classifiers, which were ANNs as well. It is important to emphasize that ANNs can easily over fit when using many inputs. For this reason, they tested subsets from *level-0* as inputs to the *level-1*. Selecting base-estimators, they used the error from a cross-validation method, using 15 folds, where only the bests were selected. We discuss this kind of feature selection and its impacts in the Section 5.3.

5 RESEARCH OPPORTUNITIES AND DISCUSSIONS

Although stacking is not a recent method, computational evolution has provided new research opportunities. In this section, we discuss some relevant issues, adding our points of view and pointing out advantages and disadvantages when using stacking.

5.1 About the Level-0

Most of works have used, in our opinion, few estimators in the *level-0*. According to preliminary experiments, we strongly believe that, with the increase of the computing power that has been happening, researchers should try the usage of hundreds of estimators in this level in order to improve the quality of their meta-models. By now, in academic works, authors have reported a small number of base-estimators (an amount around 7 according to our reading). We also performed experiments and it was very clear for us that the increase of type of models, and the variability in their parameters, can help in two ways: improving the metric that is being used, e.g., AUC, logloss, rmse, and it is also able to stabilize the results, reducing the standard deviation of the used metrics. For this reason, we recommend this attempt in future investigations.

5.2 About the Level-1

In this study, we could notice that few authors have included original features along with *level-0* meta-features to create the meta-dataset. Most of them focused on simply using predictions probabilities distribution as *level-1* dataset structure, even that this approach could be benefited from the basic features. The use of basic features from *level-0* along with meta-features can lead the meta-estimator to more accurate results, according to our preliminary experiments. This approach allows the *level-1* model to make inferences using the complete data, which includes basic and meta-features.

Besides the non-usage of basic features at *level-1*, there is little effort on stacking more than two layers, since few research explores this stacking's possibility. Each stacking's level seeks to reduce the previous level error, increasing its accuracy. In this manner, more levels can lead the framework to more precise results. Using more than 2 layers, as well as more meta-learners, have not been extensively study in the stacking research area yet and we believe it is a good opportunity for scientific investigation.

5.3 Meta-feature selection

Meta-feature selection, or model selection, to build the meta-level has been a recurrent topic in stacking research. We have presented works that have used cross-validation approaches and even optimization methods to perform this task. However, such approaches lead stacking to take more time to execute than its general framework. Selecting and removing weak estimators from *level-0* can implies on loss of performance the meta-learner, as stacking is able to take advantage even from weak *level-0* estimators and not simply from strong ones [Ozay and Yarman-Vural 2016].

Researchers have been trying to reduce the number of *level-0* estimators in order to avoid the curse of dimensionality at *level-1*. In contrast, recent machine learning algorithms already aims to reduce overfitting when learning on huge datasets. In this way, we strongly believe that we should relax meta-features selection constraints and increase meta-learner's responsibility when generating predictions. This approach takes advantage of the variety of a lot of base-estimators been combined. We consider that increase the responsibility of the meta-learning with more outputs to combine as a research opportunity.

5.4 About the k-fold generating artificial features

Note that the most common procedure, described in the Section 2.1.3, needs $p \times k$ distinct training processes to generate all artificial features, which will be used by at least one meta-learner in the *level-1*. It demonstrates one reason why this technique was not so popular some years ago. Nowadays, being able to learn faster, using several CPU and GPU cores, it is easier than last decades. Besides that, there is the requirement to run the meta-learner in the *level-1*, which is another time disadvantage when compared to regular ensemble methods. However, the additional computing cost can provide better results. Based on this, stacking has been becoming more popular since the recent technology advances and, in our opinion, it tends to be more common.

Even though creating artificial features using this procedure avoids overfitting, it presents a disadvantage, in our opinion, because of the usage of different instantiated models for each generated meta-feature $\hat{y}_i$. As shown in Section 2.1.3, in a 3-fold procedure, 3 models would be generated using different portions of the training data, and its predictions would be then concatenated into one new meta-feature. Different instantiated models at *level-0* could insert noise in the artificial feature, affecting *level-1's* generalization process. Generating meta-features with another procedure could reduce this noise and we consider it is a research opportunity and can be further investigated.

5.5 Advantages and disadvantages of using stacking

Stacking has shown itself more precise than the best single model from *level-0*, outperforming it in most literature research in this field. Besides that, as this technique can take advantage from the any *level-0* model at *level-1*. Because of this, model selection procedures can be reduced, increasing its automation, which is one of Artificial Intelligence's main goals.

Even that stacking can perform better than single models, it is more time consuming. Generate many models and stack many layers can lead to high overall accuracy, but in exchange, it can take a lot of time.

Differently from standard combining methods, such as voting, mean, median and so on, stacking learns to combine its base-estimators. In this manner, it takes advantage even from weak estimators and not only from strong ones. Besides that, stacking can increase its dataset and use original features from *level-0* at *level-1*. In this case, the method can be benefitted from context analysis, but again, can takes long to run.

6 CONCLUSIONS

In this work we have presented an overview about the Stacked Generalization method, considering different architectures and applications. For considering the potential of this technique and the amount of open questions, we have also presented arguments in order to point out why stacking is a fertile field of research, which provides several opportunities of investigation.

To sum up, nowadays, the process of creating a new decision maker takes a long time from people, mainly when they are validating their models. Stacking can transfer this hard work from humans to machines if they use several of different estimators and parameterizations in the *level-0*, and a good generalizer in the *level-1*. Besides that, once computing power is not a critical problem and the result tends to improve, the use of the technique becomes more and more attractive. After making these statements, we ask: *Is Stacked Generalization an inevitable step for the future of supervised learning?*

REFERENCES

Fatai Anifowose, Jane Labadin, and Abdulazeez Abdulraheem. 2015. Improving the prediction of petroleum reservoir characterization with a stacked generalization ensemble model of support vector machines. *Applied Soft Computing* 26 (2015), 483–496.

Sylvain Arlot, Alain Celisse, et al. 2010. A survey of cross-validation procedures for model selection. *Statistics surveys* 4 (2010), 40–79.

Weiquan Bao, Ke Chen, and Huisheng Chi. 1998. An HMM/MFNN hybrid architecture based on stacked generalization for speaker identification. In *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, Vol. 1. IEEE, 367–371.

Leo Breiman. 1996a. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

Leo Breiman. 1996b. Stacked regressions. *Machine learning* 24, 1 (1996), 49–64.

Philip K Chan and Salvatore J Stolfo. 1993. Experiments on multistrategy learning by meta-learning. In *Proceedings of the second international conference on information and knowledge management*. ACM, 314–323.

Jin Chen, Cheng Wang, and Runsheng Wang. 2009. Using stacked generalization to combine SVMs in magnitude and shape feature spaces for classification of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing* 47, 7 (2009), 2193–2205.

Yijun Chen, Man-Leung Wong, and Haibing Li. 2014. Applying Ant Colony Optimization to configuring stacking ensembles for data mining. *Expert Systems with Applications* 41, 6 (2014), 2688–2702.

Thomas G Dietterich. 1997. Machine-learning research: Four current directions. *AI magazine* 18, 4 (1997), 97–136.

Michael Doumpos and Constantin Zopounidis. 2007. Model combination for credit risk assessment: A stacked generalization approach. *Annals of Operations Research* 151, 1 (2007), 289–306.

Sašo Džeroski and Bernard Ženko. 2002. Stacking with multi-response model trees. In *International workshop on multiple classifier systems*. Springer, 201–211.

Saso Džeroski and Bernard Ženko. 2004. Is combining classifiers with stacking better than selecting the best one? *Machine learning* 54, 3 (2004), 255–273.

Asif Ekbal and Sriparna Saha. 2013. Stacked ensemble coupled with feature selection for biomedical entity extraction. *Knowledge-Based Systems* 46 (2013), 22–32.

D Fan, Sal Stolfo, and Phillip Chan. 1999. Using conflicts among multiple base classifiers to measure the performance of stacking. In *Proceedings of the ICML-99 Workshop on Recent Advances in Meta-Learning and Future Work*. Citeseer, 10–17.

Yoav Freund and Robert E Schapire. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*. Springer, 23–37.

Joao Gama. 1998. Combining classifiers by constructive induction. In *European Conference on Machine Learning*. Springer, 178–189.

João Gama and Pavel Brazdil. 2000. Cascade generalization. *Machine Learning* 41, 3 (2000), 315–343.

Ali A Ghorbani and Kiarash Owrangh. 2001. Stacked generalization in neural networks: generalization on statistically neutral problems. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, Vol. 3. IEEE, 1715–1720.

Charles Gomes, Hisham Nocairi, Marie Thomas, Fabien Ibanez, Jean-François Collin, and Gilbert Saporta. 2012. Stacking prediction for a binary outcome. In *Compstat 2012*. 271–282.

Anjali Gupta and Amit R Thakkar. 2014. Optimization of stacking ensemble Configuration based on various metahueristic algorithms. In *Advance Computing Conference (IACC), 2014 IEEE International*. IEEE, 444–451.

Michael Y Hu and Christos Tsoukalas. 2003. Explaining consumer choice through neural networks: The stacked generalization approach. *European Journal of Operational Research* 146, 3 (2003), 650–660.

Gordon Hughes. 1968. On the mean accuracy of statistical pattern recognizers. *IEEE transactions on information theory* 14, 1 (1968), 55–63.

Anna Jurek, Yaxin Bi, Shengli Wu, and Chris Nugent. 2014. A survey of commonly used ensemble-based classification techniques. *The Knowledge Engineering Review* 29, 5 (2014), 551–581.

Michael A King, Alan S Abrahams, and Cliff T Ragsdale. 2015. Ensemble learning methods for pay-per-click campaign management. *Expert Systems with Applications* 42, 10 (2015), 4818–4829.

John F Kolen and Jordan B Pollack. 1991. Back propagation is sensitive to initial conditions. In *Advances in neural information processing systems*. 860–867.

Yehuda Koren. 2009. The bellkor solution to the netflix grand prize. *Netflix prize documentation* 81 (2009), 1–10.

Agapito Ledezma, Ricardo Aler, and Daniel Borrajo. 2002. Heuristic search-based stacking of classifiers. *Heuristics and Optimization for Knowledge Discovery* 54 (2002).

Agapito Ledezma, Ricardo Aler, Araceli Sanchis, and Daniel Borrajo. 2010. GA-stacking: Evolutionary stacked generalization. *Intelligent Data Analysis* 14, 1 (2010), 89–119.

Christiane Lemke, Marcin Budka, and Bogdan Gabrys. 2015. Metalearning: a survey of trends and technologies. *Artificial intelligence review* 44, 1 (2015), 117–130.

Eitan Menahem, Lior Rokach, and Yuval Elovici. 2009. Troika–An improved stacking schema for classification tasks. *Information Sciences* 179, 24 (2009), 4097–4122.

Christopher J Merz. 1999. Using correspondence analysis to combine classifiers. *Machine Learning* 36, 1-2 (1999), 33–58.

Josef Moudrík and Roman Neruda. 2015. Evolving non-linear stacking ensembles for prediction of go player attributes. In *Computational Intelligence, 2015 IEEE Symposium Series on*. IEEE, 1673–1680.

Steven R Ness, Anthony Theocharis, George Tzanetakis, and Luis Gustavo Martins. 2009. Improving automatic music tag annotation using stacked generalization of probabilistic svm outputs. In *Proceedings of the 17th ACM international conference on Multimedia*. ACM, 705–708.

Hicham Noçairi, Charles Gomes, Marie Thomas, and Gilbert Saporta. 2016. Improving Stacking Methodology for Combining Classifiers; Applications to Cosmetic Industry. *Electronic Journal of Applied Statistical Analysis* 9, 2 (2016), 340–361.

Nikunj C Oza and Kagan Tumer. 2008. Classifier ensembles: Select real-world applications. *Information Fusion* 9, 1 (2008), 4–20.

Mete Ozay and Fatos Tunay Yarman-Vural. 2016. Hierarchical distance learning by stacking nearest neighbor classifiers. *Information Fusion* 29 (2016), 14–31.

Robi Polikar. 2006. Ensemble based systems in decision making. *IEEE Circuits and systems magazine* 6, 3 (2006), 21–45.

Georgios Sakkis, Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Constantine D Spyropoulos, and Panagiotis Stamatopoulos. 2001. Stacking classifiers for anti-spam filtering of e-mail. *arXiv preprint cs/0106040* (2001).

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)* 34, 1 (2002), 1–47.

Alexander K Seewald. 2002. How to make stacking better and faster while also taking care of an unknown weakness. In *Proceedings of the nineteenth international conference on machine learning*. Morgan Kaufmann Publishers Inc., 554–561.

M Paz Sesmero, Agapito I Ledezma, and Araceli Sanchis. 2015. Generating ensembles of heterogeneous classifiers using stacked generalization. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 5, 1 (2015), 21–34.

P Shunmugapriya and S Kanmani. 2013. Optimization of stacking ensemble configurations through Artificial Bee Colony algorithm. *Swarm and Evolutionary Computation* 12 (2013), 24–32.

Joseph Sill, Gábor Takács, Lester Mackey, and David Lin. 2009. Feature-weighted linear stacking. *arXiv preprint arXiv:0911.0460* (2009).

Kai Ming Ting and Ian H Witten. 1997a. Stacked Generalization: when does it work? (1997).

Kai Ming Ting and Ian H Witten. 1997b. Stacking bagged and dagged models. (1997).

Kai Ming Ting and Ian H Witten. 1999. Issues in stacked generalization. *J. Artif. Intell. Res.(JAIR)* 10 (1999), 271–289.

Ljupčo Todorovski and Sašo Džeroski. 2000. Combining multiple models with meta decision trees. In *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 54–64.

Ljupčo Todorovski and Sašo Džeroski. 2003. Combining classifiers with meta decision trees. *Machine learning* 50, 3 (2003), 223–249.

Joaquín Torres-Sospedra, Carlos Hernández-Espinosa, and Mercedes Fernández-Redondo. 2006. Combining MF networks: A comparison among statistical methods and stacked generalization. *ANNPR* 4087 (2006), 210–220.

Andreas Töscher, Michael Jahrer, and Robert M Bell. 2009. The bigchaos solution to the netflix grand prize. *Netflix prize documentation* (2009), 1–52.

George Tzanis, Christos Berberidis, and Ioannis Vlahavas. 2007. MANTIS: a data mining methodology for effective translation initiation site prediction. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*. IEEE, 6343–6347.

George Tzanis, Christos Berberidis, and Ioannis Vlahavas. 2012. StackTIS: A stacked generalization approach for effective prediction of translation initiation sites. *Computers in biology and medicine* 42, 1 (2012), 61–69.

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

Ian H Witten, Eibe Frank, and Mark A Hall. 2011. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc.

David H Wolpert. 1992. Stacked generalization. *Neural networks* 5, 2 (1992), 241–259.

Wanli Xing, Xin Chen, Jared Stein, and Michael Marcinkowski. 2016. Temporal prediction of dropouts in MOOCs: Reaching the low hanging fruit through stacking generalization. *Computers in Human Behavior* 58 (2016), 119–129.

Bernard Ženko, Ljupčo Todorovski, and Sašo Džeroski. 2001. *A comparison of stacking with MDTs to bagging, boosting, and other stacking methods*.

Dan Zhu. 2010. A hybrid approach for efficient ensembles. *Decision Support Systems* 48, 3 (2010), 480–487.