The Astrophysical Journal, 819:18 (11pp), 2016 March 1 doi:10.3847/0004-637X/819/1/18

META-CLASSIFICATION FOR VARIABLE STARS

Karim Pichara[1,2,3], Pavlos Protopapas[2], and Daniel León[1]
[1] Computer Science Department, Pontificia Universidad Católica de Chile, Santiago, Chile
[2] Institute for Applied Computational Science, Harvard University, Cambridge, MA USA
[3] Millennium Institute of Astrophysics, Chile
Received 2015 November 13; accepted 2016 January 12; published 2016 February 24

ABSTRACT

The need for the development of automatic tools to explore astronomical databases has been recognized since the inception of CCDs and modern computers. Astronomers already have developed solutions to tackle several science problems, such as automatic classification of stellar objects, outlier detection, and globular clusters identification, among others. New scientific problems emerge, and it is critical to be able to reuse the models learned before, without rebuilding everything from the beginning when the sciencientific problem changes. In this paper, we propose a new meta-model that automatically integrates existing classification models of variable stars. The proposed meta-model incorporates existing models that are trained in a different context, answering different questions and using different representations of data. A conventional mixture of expert algorithms in machine learning literature cannot be used since each expert (model) uses different inputs. We also consider the computational complexity of the model by using the most expensive models only when it is necessary. We test our model with EROS-2 and MACHO data sets, and we show that we solve most of the classification challenges only by training a meta-model to learn how to integrate the previous experts.

Key words: methods: data analysis – stars: statistics – stars: variables: general – surveys

1. INTRODUCTION

The scientific community is dealing with massive amounts of digital information and astronomy is not an exception (see, for example, Cook et al. 1995; Derue et al. 2002; Kaiser 2004; Ivezic et al. 2008; Udalski et al. 2008). It is practically impossible to analyze the vast amount of data, generated by modern telescopes and surveys, without the help of machines. This is done either with the use of simple algorithmic solutions or machine learning approaches. A particular example of such automatic methods is the automatic classification of variable objects (Bloom & Richards 2011; Bloom et al. 2011; Richards et al. 2011; Kim et al. 2012; Pichara et al. 2012; Pichara & Protopapas 2013), which is the focus of this paper. Automatic classification of variable stars makes it possible to speed scientific discoveries through an initial labelling, thus allowing astronomers to have a selection of light curves of interest for further study and analysis. Many solutions in automatic classification have been proposed (Butler & Bloom 2011; Richards et al. 2011; Long et al. 2012; Pichara et al. 2012; Kim et al. 2014). These models, called *experts* in this paper, classify to a subset of possible classes, using a set of specific variables (hereinafter called features) that represent the light curves. In this work, we suggest that future models can take advantage of those models in solving new challenges. As an example, suppose that we have the following models.

1. A model that classifies objects in quasars and no-quasars, trained with a specific set of features (Kelly et al. 2009; Pichara et al. 2012; Kim et al. 2012).
2. A model that separates periodic from non-periodic objects (Huijse et al. 2012; Protopapas et al. 2015; Kim et al. 2014).
3. A general purpose classifier that can classify (with a bit lower accuracy) many different variability classes (Richards et al. 2011; Long et al. 2012; Pichara & Protopapas 2013).
4. A model that classifies RR Lyrae (Gran et al. 2015) from the rest.
5. A model that identifies microlensing and eclipsing binaries (Belokurov et al. 2003).

Assuming we need to create a model that classifies RR Lyrae, Eclipsing Binaries, Be stars, and quasars, it is apparent that there is a lot of intersection between the new desired model and the previous models we have. Therefore, we should be able to solve our new challenge without the need to build a totally new model.

The idea of mixing many different models is very old in machine learning literature (Rasmussen & Ghahramani 1991; Jordan & Jacobs 1993; Meir 1996; Breiman 2001; Kuncheva & Whitaker 2003; Kuncheva 2007; Bishop & Svensen 2012; Chamroukhi 2015). These approaches are guided by the "divide and conquer" principle, in which each expert focuses on a particular area of feature space. Most of the solutions proposed from machine learning literature assume the same context for each of the experts. In other words, they deal with data represented in the same feature space (same variables describing the data), and in most cases with the same set of predicted classes. To the best of our knowledge, there is not a mixture of expert solutions that combine all of the context variants we mentioned before, together with the efficient management of the computational complexity of the experts.

Our approach is based on the very simple idea of empirically estimating how fitting a model is in a given scenario. In other words, the best we can do in learning how to combine different experts is to try them in different cases and evaluate their results. This method allows us to avoid the need for the understanding of the internal structure of the experts, which can be very costly. Our approach first creates a meta-data set containing all of the experts' outputs obtained from the initial training data set. We search for patterns in the classification results and we model these patterns to predict the light-curve

classes. This is the meta-classifier, which integrates the outputs from the experts to make a final prediction.

The idea of studying model outputs has been used before, but in different contexts, such as anomaly detection (Nun et al. 2014) and measurements of diversity (Kuncheva & Whitaker 2003).

Furthermore, the integration model has to be easily understood and interpretable. It is not desirable to have a "black box" that integrates the decision in an unknown and confusing way because we cannot gain intuition on how the model is deciding or how each expert is contributing to the final decision. Decision trees are very suitable for simple decision patterns; each node represents a question, and each directed edge pointing out from a node represents an answer to the node. There are many algorithms proposed to train a decision tree (Quinlan 1986, 1993), but those algorithms just focus on optimizing the classification accuracy rather than consider the cost of each question done on each of the nodes.

There are many known techniques that aim to describe light curves as vectors of real numbers (features) by trying to extract the maximum information from light curves while maximizing the classification performance. In a recent work, Nun et al. (2015) presented an automatic tool that calculates more than 60 such features. Depending on the classification task, some features are more useful than others, and some features are more computationally costly than others. One example of computationally costly features are the coefficients for a continuous autoregressive model (Kelly et al. 2009; Pichara et al. 2012). However, these features have been shown to be helpful in differentiating quasars from other stars. Another example is the correntropy kernelized periodogram (Huijse et al. 2012), which has been used to classify periodicity classes. Both kinds of features are more expensive than others, but both present significant improvements on the classification tasks.

In our mixture of experts set up, each of the models solves different problems using different features with different estimation cost. The cost of classification from the mixture of experts can be calculated. Every time we ask a model to classify a given light curve, we know the features used by that model, and we either empirically measure or estimate the cost of each feature evaluation. Our meta-model deals with the experts' cost by minimizing the overall cost. Besides the classification accuracy of each expert, the algorithm also considers their cost.

This work is organized as follows. In Section 2, we present a brief description of the current research in mixtures of experts and related topics. In Section 3, we give all of the details of the proposed methodology. In Section 4, we describe the experimental results obtained in different tests with real data sets. Finally, in Section 5, we discuss the main results of our work.

2. RELATED WORK

There are dozens of different methodologies to improve classification rates by combining the "expertise" of different classifiers. This whole topic is known in the machine learning community as *ensemble learning* or *mixture of experts* (Jordan & Jacobs 1993; Bishop 2006). One of the earliest discussions of ensemble learning appears in the work of Bazell & Aha (2001), in which they combine different instances of the same model via bootstrapping, train each classifier in a randomly chosen sub-sample of the training set, and finally predict through majority voting.

The work of Freund et al. (1999) introduces the *Adaboost* algorithm, which combines classifiers in a cascade scheme. In the cascade of classifiers, each model trains only with the instances that the previous model predicted incorrectly. This is achieved by tuning hyperparameters that control the false positive rate and the minimum acceptable detection rate. Unfortunately, this method works only for the two class problem, though there are works that discuss extensions to multi-classes (Lin & Liu 2005; Zehnder et al. 2008).

Although this ensemble method achieves good classification rates, it cannot be applied to our problem. This is because the combinations and the classifiers are trained together; classifiers are not already trained. In our case, experts are already trained, and we do not need to train them again. On the contrary, we want to reuse previously acquired knowledge. Moreover, most boosting methods assume that instances of each model are represented through the same feature space (they use the same features on every model), and the predicted classes for every model are also the same. In our case, we use different features for each model and different output classes.

In Faraway et al. (2014), similarly to our work, one of the classifiers they consider is hierarchical. They first evaluate whether or not the object is a transient and, depending on the answer, they attempt to classify among the other classes. What makes a big difference is that we propose a model that automatically learns that hierarchy and is able to create different hierarchical classifiers depending on the case. On the other hand, Faraway et al. (2014) define a hierarchical classifier where the structure is set by hand and there is no learning process about that hierarchy.

A seminal work in the mixture of experts was proposed in Jordan & Jacobs (1994). They create a hierarchy of base level models that specialize in separate areas of the input space. On each level of the hierarchy, each expert is combined by a gate function that learns a model-combination function that varies depending on the instance to be classified. The combination function assigns a weight to each model in the final prediction. In the original paper, this function is a multinomial distribution, but there exist extensions using probability models from the exponential family (Xu et al. 1995). Unfortunately, like most of the current machine learning approaches in mixtures of experts, this method is not helpful for our proposes because the gate functions and the base models are all trained together to create the effect of specialization/cooperation and, therefore, all of the models must belong to the same problem context (features and classes).

Another perspective of ensemble modeling, in the context of meta-models, is the use of a technique called *Stacked Generalization* (Wolpert 1992). In this framework, each base model (or *level-0* model, in the nomenclature of the cited work) is "fed" with the data and the output of these models is considered to be an input for a meta-model (*level-1* model). This essentially creates an *Intermediate Feature Space* (Kuncheva 2004) where the second stage learning can be performed. In the work of Wolpert (1992), this is referred as a *level-1* data set, and the *level-0* data set is where the *level-0* models are trained. This process can be repeated an indefinite number of times. Intuitively, the meta-model objective is to correct the bias of the base models (LeBlanc & Tibshirani 1996).

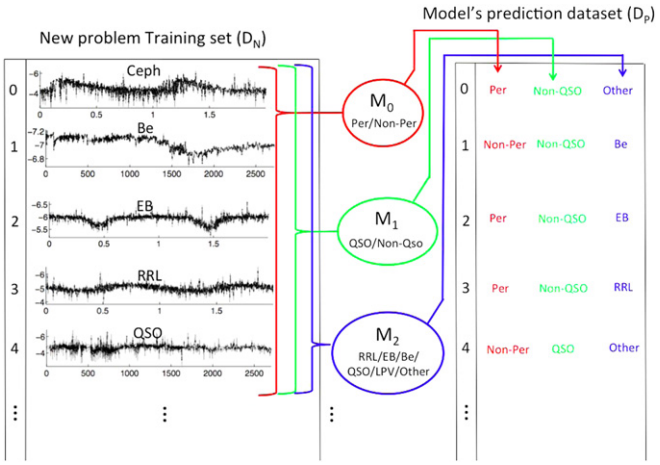

Figure 1. Graphical description of the creation of model prediction data (D_P). We can see that starting from a set of labeled light curves (D_N) we ask for each model to predict the class of each of those light curves, then we record their predictions as new rows in D_P, generating a second data set later used to train the mixture of models.

Rather than focus on work reutilization, most of the methods mentioned above concentrate on the "divide and conquer" principle and they do not consider cost, making them hard to use in the framework we are addressing in this work. Furthermore, to the best of our knowledge, there is no work in the field of astronomy addressing the problem of how to automatically combine previously learned models. We believe that in the area of light-curve classification, automatic integration can make important contributions, especially with the continuous growth of data and models.

3. PROPOSED METHOD

We start by assuming that we have m already trained models $\{M_1, M_2, \ldots, M_m\}$, where each model M_i corresponds to a light-curve classifier. Each classifier M_i uses a specific set of features F_{M_i} to represent the light curves and classifies each light curve into a set C_{M_i} of possible classes. For example, M_0 can be a model that uses the features $F_{M_0} = \{\text{Amplitude}, \text{Autocor} - \text{length}, \text{CAR} - \text{tau}, \text{FluxPercentileRatio}\}$, and is able to classify them into $C_{M_i} = \{\text{QSO}, \text{RRL}, \text{Be}, \text{Other}\}$. Besides having the already trained models $\{M_1, M_2, \ldots, M_m\}$, we have a training set (D_N), corresponding to the data associated with the new classification problem, the one we need to solve with the trained models.

3.1. Creation of the Prediction Data Set

The first step of the process is to create a (meta-)data set containing the predictions of each model (D_P) associated to each of the light curves in D_N. The main purpose of D_P is to have training data for the meta-model. To create D_P, we just run each of the trained models, getting their predictions on the training set D_N. Then, we save those predictions as rows in D_P together with the real class label of the light curve. Figure 1 shows an example of this process for three given models.

3.2. Meta-model Representation

After obtaining D_P, we can build a meta-model that efficiently mixes the decision of each of the previously trained models. The meta-model has to act as a "director." Every time

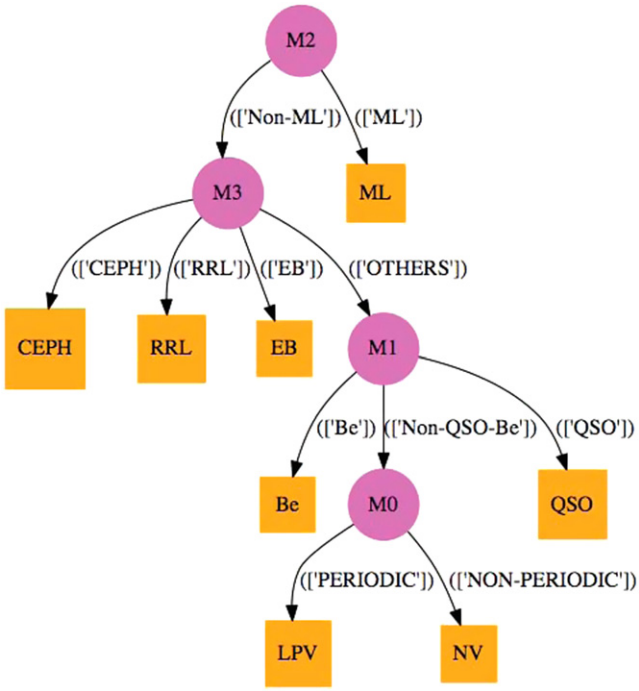

Figure 2. Example of a meta-model. Round nodes represent the previously trained models, edges show which path to follow depending on the model's prediction, and square nodes represent the leaves that correspond to a final prediction done by the meta-model.

the meta-model receives a new query light curve, it has to choose which is the first model to be used, then, depending on the prediction of that model, select the next model, and so forth. A natural representation of the meta-model is a decision tree structured schema, where each node represents one of the previously trained models M_i. Each of the edges pointing out from each node represents one of the possible predictions made from the model represented by the node, and leaves represent a final prediction done by the meta-model. Figure 2 shows an example with four models $\{M_0, M_1, M_2, M_3\}$. The tree structure meta-model first asks model M_2 to do the prediction. In the case that M_2 predicts a microlensing (*ML*), the meta-model immediately predicts *ML* (reaches a leaf). In the case that model M_2 says *Non-ML* the meta-model asks model M_3 for a prediction. If M_3 predicts Cepheid (CEPH), RR Lyrae (RRL), or Eclipsing Binary (EB) the meta-model predicts according to M_3, but in the case that M_3 predicts *OTHERS*, the meta-model now asks M_1 for a prediction, and so on. This kind of structure is very suitable for what we need. It is very easy to understand, uses a very well-known data structure from computer science (very mature searching and traversal algorithms), and the most important benefit is that it can be interpretable.

3.3. Automatically Building the Meta-model

After understanding the structure of the meta-model and how it works, the central question is how do we build it? We propose an algorithm that is mainly driven by the probability that a given model correctly predicts the class of a light curve and the cost of running that model. The likelihood that a model correctly predicts the class of a given light curve can be estimated from the training data (D_P), and the cost of running that model can be easily calculated from the cost of all the features the model uses to represent the light curve.

The meta-model learning algorithm is inspired by the classical decision tree learning algorithm (Quinlan 1986, 1993). Given a score that measures the quality of any node, the best node is selected to be the root of the tree. Then, the algorithms traverse down from each of the possible edges pointing out from the root (possible predictions of the model associated to the root) and recursively searches for the next best model. We select the best model (M_*) for a given node of the tree as follows:

$$M_* = \arg\max_{M_i} \frac{\text{Info\_Gain}(M_i)}{E\left[\text{Cost}(M_i)\right]},\ i \in [1,\ldots,m], \tag{1}$$

where $\text{Info\_Gain}(M_i)$ is the information gain (Quinlan 1986) of model M_i, which measures the expected reduction in entropy in D_P when model M_i makes a prediction. It is defined as

$$\begin{aligned}\text{Info\_Gain}(M_i) &= H(\text{class}) - \sum_{v \in C_{M_i}} \frac{|v|}{|C_{M_i}|} H(\text{class}|v)\\ H(\text{class}) &= -\sum_{k \in C^M} \frac{|k|}{|C^M|} \log \frac{|k|}{|C^M|}\\ H(\text{class}|v) &= -\sum_{k \in C_v^M} \frac{|k|}{|C_v^M|} \log \frac{|k|}{|C_v^M|},\end{aligned} \tag{2}$$

where C^M is the union of all possible classes predicted among all models. Similarly, C_v^M is the union of all classes predicted across the models $\{M_1, M_2, \ldots, M_{i-1}, M_{i+1}, \ldots, M_m\}$ when the model M_i predicts v. In simpler words, $H(\text{class}|v)$ is the entropy of the class column of D_P selecting only the rows of D_P that match $M_i = v$. Intuitively, the information gain tells us if a model's M_i predictions are good enough to separate among possible classes, in the sense that if every time we instantiate the model M_i to its possible predictions, we see whether the uncertainty in the class column is reduced or not (entropy). This concept is directly related to the probability of getting a successful classification if the meta-model uses M_i to do the final prediction.

The term $E\left[\text{Cost}(M_i)\right]$ is the expected cost of a model, estimated as follows:

$$\begin{aligned}E\left[\text{Cost}(M_i)\right] = P_L(M_i)\text{Cost}(M_i) + (1 - P_L(M_i))\\ \times \left[\sum_{v \in C_{M_i}} \sum_{j=i+1}^{m} P_L(M_j|M_i = v)\right.\\ \left.\times\ \text{Cost}(M_j|M_i = v)\right].\end{aligned} \tag{3}$$

The term $P_L(M_i)$ indicates the probability that model M_i reaches a leaf in the tree in the next step. In other words, how likely it is that model M_i will be making a final decision (reaching a leaf). Given that the decision tree algorithm creates a leaf every time most of the remaining instances belong to the same class, to estimate the probability of reaching a leaf, we need an indicator of how good the model was after predicting a given class. This is also related to the information gain of the model at that level of the tree. To have valid probability values, we normalize the information gain from [0, 1] as

$$P_L(M_i) \approx 1 - \frac{\sum_{v \in C_{M_i}} \frac{|v|}{|C_{M_i}|} H(\text{class}|v)}{H(\text{class})}. \tag{4}$$

The cost of model M_i ($\text{Cost}(M_i)$) is calculated as the sum of the features that model M_i uses to represent each light curve. The second part of Equation (3) is basically the weighted sum of every model, except for model M_i cost, where each weight corresponds to the probability that the given model reaches a correct leaf in the tree in the next step. Intuitively, Equation (1) is finding the model whose cost is minimum and, at the same time, taking the meta-model models to the right prediction.

We summarize the training and predicting steps of the meta-classification process below.

Training:

1. For each new model M_i, create a new data column $D_P[i]$ with the prediction of each model M_i over the training data.
2. Build D_P as a union of all the predictions, $D_P = \bigcup_{i=1}^{m} D_P[i]$.
3. Create the meta-training set, adding to D_P a column with the known class of each object (this is the same class column included in D_N).
4. Build the meta-model according to Section 3.3.

Predicting:

1. For any unclassified light curve x, start traversing the meta-model tree from the root.
2. On each node M_i, extract the features F_{M_i}, go down the tree according to the prediction of M_i until a leaf is reached.
3. Predict according to the reached leaf.

4. EXPERIMENTAL RESULTS

We tested our model with two light-curve data sets, MACHO (Cook et al. 1995) and EROS-2 (Tisserand et al. 2007). On each data set, we created different expert models trained to classify different subsets of variability classes. Each model in the setup uses a specific set of features to describe the light curves. These specific sets are determined using a feature importance algorithm called *mean decrease impurity*, described in Breiman et al. (1984). After a particular model M_i is trained, if the meta-model requires a prediction from M_i, it will only extract the features included on M_i's specific set. Note that if another model previously extracted some of the required features, they will not be extracted again.

For each of the experts, we use a Random Forest classifier (Breiman 2001). We use the FATS (Feature Analysis for Time Series; Nun et al. 2015) tool to extract the features of light curves. This tool is able to extract up to 64 different features per lightcurve. All details about the meaning of each of the features can be found in Nun et al. (2015). As mentioned above, some of the features are more expensive than others. Since each expert uses a selection of the best features according to its own classification problem, models have different associated costs.

All of the accuracy results are presented throughout recall, precision, f-score, and confusion matrix. All of these indicators

Table 1
Number of Instances per Class of Variability in the MACHO Training Set

Class	# Instances
Be stars	127
CEPH	101
EB	255
LPV	361
ML	580
NV	3963
QSO	59
RRL	613

were obtained using a 10-fold cross-validation process on each of the training sets.

4.1. MACHO Data Set

The MACHO Project (Massive Compact Halo Objects; Cook et al. 1995) observed the Magellanic Clouds and Galactic bulge with the main purpose of detecting microlensing events. Observations were done using blue (~4500–6300 Å) and red (~6300–7600 Å) passbands. The cadence is about one observation per two days for 7.4 years, which generates approximately 1000 observations per object. The light curves used in this work are from the Small and Large Magellanic Clouds. The fields cover almost the entire LMC bar (10 square degrees) to a limiting magnitude of $V \approx 22$. The training set contains 6059 labeled light curves (Kim et al. 2011). Table 1 shows the number of light curves per each of the available classes. We created seven models to work as experts, each one

Table 3
Accuracy Indicators per Each Class on Each of the Models' Problems in the MACHO Data Set

Model	Class	Precision	Recall	F-score
M6	Be	0.733	0.780	0.756
	OTHERS	0.985	0.985	0.985
	ML	0.970	0.964	0.967
	EB	0.877	0.867	0.872
M0	PERIODIC	0.957	0.962	0.960
	NON-PERIODIC	0.989	0.988	0.989
M3	Non-ML	0.995	0.997	0.996
	ML	0.970	0.957	0.964
M2	Be	0.866	0.811	0.837
	QSO	0.738	0.525	0.614
	Non-QSO-Be	0.994	0.998	0.996
M4	CEPH	0.929	0.901	0.915
	RRL	0.967	0.949	0.958
	EB	0.900	0.878	0.889
	OTHERS	0.993	0.997	0.995
M1	NON-QSO	0.995	0.998	0.996
	QSO	0.675	0.458	0.545
M5	CEPH	0.936	0.871	0.903
	OTHERS	0.954	0.976	0.965
	EB	0.897	0.855	0.876
	NV	0.992	0.987	0.990

Table 2
Pretrained Models for MACHO Data Set, Features Used on each Model, Classes That Each Model Can Predict, and Cost That Each Model Takes To Represent One Light Curve

Name	Features Used in the Model	Possible Classes	Avg. Cost per Light curve (secs)
M_0	Psi eta, StetsonL, Psi CS, PeriodLS, StetsonJ, Rcs, Period fit, StetsonK AC	PERIODIC, NON-PERIODIC	1.729
M_1	Rcs, Color, PeriodLS, Psi CS, Auto-cor-length, Mean, MedianAbsDev, StetsonJ, CAR tau, CAR mean, StetsonL, PercentDifferenceFluxPercentile, Q31, SlottedA length, Eta e, AndersonDarling, Con, FluxPercentileRatioMid65, Freq1 harmonics rel phase 1, Q31 color, Freq2 harmonics amplitude 2, Meanvariance, MedianBRP, Skew, MaxSlope	NON-QSO, QSO	2.554
M_2	Rcs, PeriodLS, Color, Autocor length, Psi CS, SlottedA length, StetsonL, Meanvariance, StetsonJ, PercentAmplitude, Amplitude, Std, Mean, Psi eta, CAR tau, FluxPercentileRatioMid65, Con, Freq3 harmonics amplitude 0	Non-QSO-Be, Be, QSO	2.551
M_3	Color, Con, SlottedA length, Mean, Rcs, StetsonK, Eta e, Skew	Non-ML, ML	0.823
M_4	Psi eta, PeriodLS, Rcs, Psi CS, CAR mean, StetsonL, CAR tau, Period fit, StetsonJ, FluxPercentileRatioMid35, Skew, Mean, Color	CEPH, RRL, EB, OTHERS	1.730
M_5	Psi eta, SlottedA length, Psi CS, StetsonJ, Color, StetsonL, Period fit, StetsonK AC, Con, Rcs, FluxPercentileRatioMid35, FluxPercentileRatioMid50, Eta e, Skew, Beyond1Std, FluxPercentileRatioMid80, FluxPercentileRatioMid65, FluxPercentileRatioMid20, PeriodLS, MedianBRP	CEPH, OTHERS, NV, EB	2.550
M_6	Color, Rcs, Skew, SlottedA length, Con, Psi CS, Psi eta, StetsonJ, PeriodLS, Eta e, StetsonK, FluxPercentileRatioMid35, Mean, Period fit, CAR mean, StetsonL, FluxPercentileRatioMid50, FluxPercentileRatioMid20, FluxPercentileRatioMid65, CAR tau, Autocor length, Q31 color, Beyond1Std	EB, OTHERS, Be, ML	2.552

Note. The cost is directly related to the features that models must extract in order to classify a given light curve.

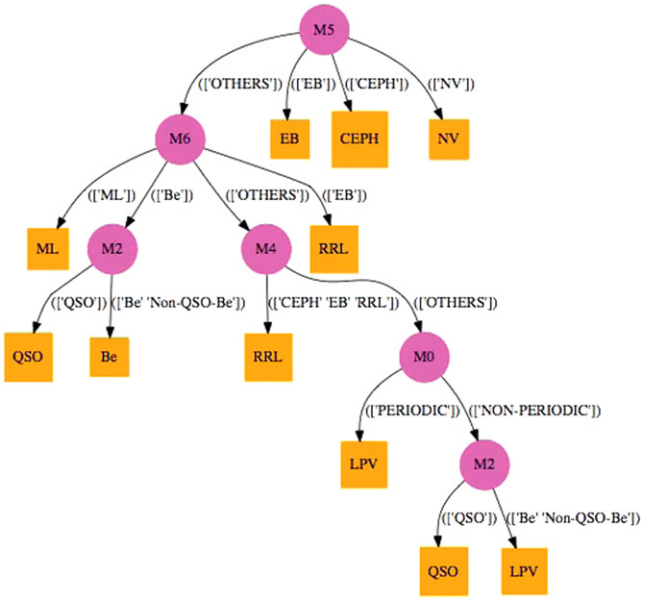

Figure 3. Meta-model learned from the MACHO training set.

Table 4
Accuracy Indicators per Class for the Meta-model in the MACHO Training Set

Class	Precision	Recall	F-Score
Be	0.857	0.756	0.803
CEPH	0.936	0.871	0.903
EB	0.897	0.855	0.876
LPV	0.799	0.978	0.879
ML	0.977	0.960	0.969
NV	0.992	0.987	0.990
QSO	0.732	0.508	0.600
RRL	0.946	0.949	0.948

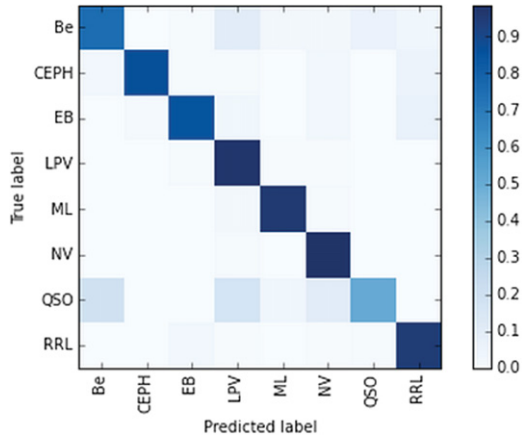

Figure 4. Confusion matrix for the meta-model learned from the MACHO training set.

trained on a specific problem, with a specific set of features. Table 2 shows the features used on each model and the classes each model predicts.

Table 3 presents the precision, recall, and f-score of each of the classes per each of the models. Most of the models are getting high f-scores for all their classes. We can see that quasars are the most complicated objects, mainly because they are confused with Be stars (model *M*2).

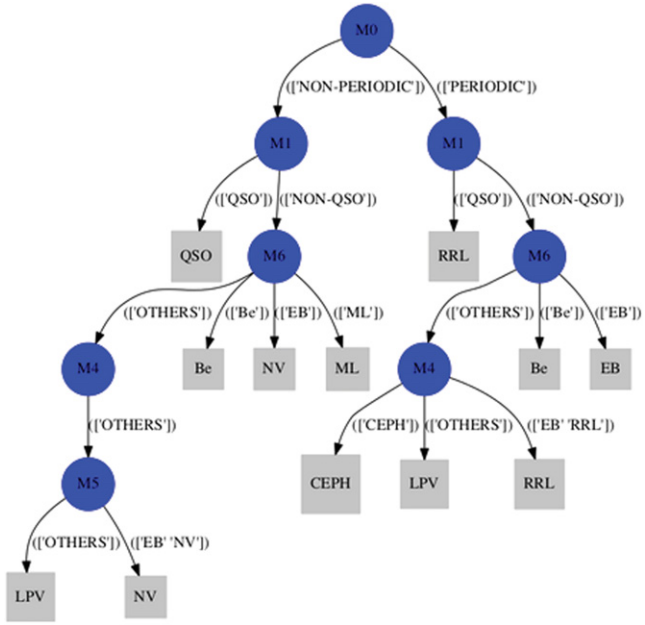

Figure 5. Meta-model learned from the MACHO training set without considering the cost of the models.

Table 5
Accuracy Indicators per each Class for the Meta-model without Considering the Cost of Models in MACHO Training Set

Class	Precision	Recall	F-Score
Be	0.832	0.740	0.783
CEPH	0.938	0.901	0.919
EB	0.884	0.863	0.873
LPV	0.779	0.978	0.867
ML	0.974	0.964	0.969
NV	0.993	0.985	0.989
QSO	0.667	0.441	0.531
RRL	0.954	0.938	0.946

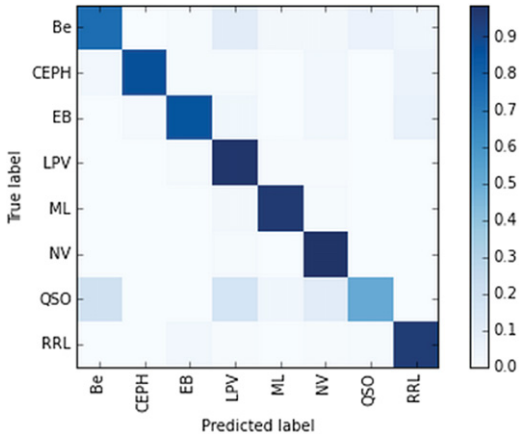

Figure 6. Confusion matrix for the meta-model without considering the cost of the experts, learned from the MACHO training set.

After learning the meta-model from the MACHO data using the proposed algorithm, we obtained the structure that is shown in Figure 3. We can see how the meta-model performs the classification. The meta-model starts by asking *M*5 and if *M*5 predicts "EB," "CEPH," or "NV," the meta-model predicts as *M*5 without asking any other model, but if *M*5 predicts

Table 6
Number of Instances per Class of Variability in EROS Training Set

Class	# Instances
Ceph 1O	870
Ceph F	1272
Ceph 1O 2O	111
EB	13523
LPV OSARG RGB O	31487
LPV SRV AGB O	4337
LPV SRV AGB C	3748
LPV Mira AGB C	760
LPV Mira AGB O	320
RRL	12167
T2CEPH	123

"Others," then the meta-model asks for the prediction from $M6$, and so on. From the tree, we can also see in most cases that the meta-model asks other models when the prediction is not so confident, like "Others" or when there is a hard class. For example, if $M6$ says that the object is a Be star, the meta-model does not predict immediately, but it continues and asks $M2$, which also knows about Be stars, and $M2$ predicts a quasar (which is usually confused with Be stars); the meta-model also predicts a Quasar. If $M2$ predicts a Be star, the meta-model can also predict a Be star. A more interesting situation occurs when $M2$ predicts "Non-QSO-Be" (Non-Quasar and Non-Be star). In this case, the best decision the meta-model can make is to predict a Be star, which is more likely than any other class given that $M6$ predicted a Be star. It is also interesting to see that the meta-model can classify Long Period Variables (LPV) even if none of the previous models can classify them, mainly because from the training set the meta-model could infer prediction patterns from the models that occur together with the LPV class. From the tree, we can see that the meta-model is predicting LPV by discarding the other classes because most of the edges along the paths that end up in an LPV tree correspond to predictions for "Others" or "Non-<some classes>" from most of the models. Another very fascinating pattern happens with LPV; the meta-model realizes that LPV is a Periodic star, after the model $M0$ in the fourth level of the tree. Note also that the meta-model does not use model $M1$. Model $M1$ classifies between Non-QSO and QSO, which are classes already covered by model $M2$.

To show that the meta-model does not sacrifice performance after the integration, Table 4 shows recall, precision, and f-score of the final meta-model. In Figure 4, we can see the confusion matrix of the meta-classifier. Most of the recall, precision, and f-score values are maintained in the meta-classifier, even some indicators are improving, such as in the case of Cepheids, as a result of the collaboration of two different models that are able to classify Cepheids.

Table 7
Pretrained Models for the EROS Data Set, Features Used on each Model, Classes that each model can Predict, and Cost that each Model takes to Represent One Light Curve

Name	Features Used in the Model	Possible Classes	Avg. Cost Per Light curve (secs)
M_0	Color, Mean, PeriodLS, CAR mean, Q31 color, Autocor length, CAR tau, FluxPercentileRatioMid50, FluxPercentileRatioMid35, SlottedA length, FluxPercentileRatioMid65, Rcs, Q31, MedianAbsDev, FluxPercentileRatioMid20, Beyond1Std, CAR sigma, Amplitude, Psi eta	OTHERS, CEPHEID, RRL, EB	16.326
M_1	Color, Mean, CAR mean, Autocor length, Q31 color, CAR tau, SlottedA length, PeriodLS, Rcs, CAR sigma, Amplitude, PercentDifferenceFluxPercentile	LPV, Non-LPV	16.325
M_2	Color, Mean, PeriodLS, CAR mean, Q31 color, CAR tau, FluxPercentileRatioMid50, SlottedA length, FluxPercentileRatioMid35, Autocor length, FluxPercentileRatioMid65, FluxPercentileRatioMid20, CAR sigma, Rcs, Q31, MedianAbsDev, Beyond1Std, Amplitude, Freq1 harmonics amplitude 0, Psi eta	OTHERS, RRL, CEPH T2CEPH, EB	21.644
M_3	PeriodLS, Mean, Color, FluxPercentileRatioMid50, Q31, FluxPercentileRatioMid35, Q31 color, MedianAbsDev, CAR mean, FluxPercentileRatioMid65, CAR tau, Freq1 harmonics amplitude 0, FluxPercentileRatioMid20, Beyond1Std, Std, StetsonK, Rcs, SlottedA length, StetsonL, FluxPercentileRatioMid80, PercentDifferenceFluxPercentile, Amplitude	OTHERS, Ceph 1O 2O, RRL, Ceph 1O, T2CEPH, Ceph F	7.866
M_4	Color, Mean, Q31 color, SlottedA length, PercentDifferenceFluxPercentile, CAR mean, Amplitude, Autocor length, Q31, Std, CAR tau, MedianAbsDev, Meanvariance, StetsonJ, Freq1 harmonics amplitude 0, PeriodLS, Rcs	OTHERS, LPV OSARG RGB O, LPV Mira AGB C, LPV SRV AGB O, LPV SRV AGB C, LPV Mira AGB O	7.866
M_5	PeriodLS, Mean, Color, FluxPercentileRatioMid50, FluxPercentileRatioMid35, MedianAbsDev, Q31 color, Q31, CAR mean, Freq1 harmonics amplitude 0, FluxPercentileRatioMid20, FluxPercentileRatioMid65, CAR tau, Beyond1Std, Autocor length, FluxPercentileRatioMid80, SlottedA length, StetsonK, Std, Skew, StetsonL	OTHERS, CEPHEID, RRL	7.867

Note. The cost is directly related to the features that models must extract in order to classify a given light curve.

Table 8
Accuracy Indicators per Class on Each of the Model Problems in the EROS Data Set

Model	Class	Precision	Recall	F-score
M5	RRL	0.960	0.938	0.949
	OTHERS	0.985	0.991	0.988
	CEPHEID	0.962	0.952	0.957
M1	LPV	0.992	0.996	0.994
	Non-LPV	0.994	0.988	0.991
M0	RRL	0.965	0.937	0.951
	CEPHEID	0.957	0.957	0.957
	OTHERS	0.992	0.995	0.993
	EB	0.939	0.954	0.946
M3	RRL	0.957	0.938	0.947
	T2CEPH	0.944	0.545	0.691
	Ceph 1O 2O	0.758	0.676	0.714
	Ceph 1O	0.926	0.860	0.892
	Ceph F	0.965	0.965	0.965
	OTHERS	0.984	0.991	0.988
M2	RRL	0.966	0.938	0.952
	CEPH T2CEPH	0.988	0.948	0.968
	OTHERS	0.992	0.997	0.994
	EB	0.940	0.956	0.948
M4	LPV Mira AGB C	0.872	0.867	0.869
	LPV SRV AGB C	0.947	0.921	0.934
	LPV OSARG RGB O	0.974	0.989	0.982
	LPV SRV AGB O	0.901	0.863	0.882
	OTHERS	0.994	0.989	0.992
	LPV Mira AGB O	0.904	0.794	0.845

To show the contribution of the cost estimation of each model, we run the same experiment only considering the information gain in the score of each model; in other words, we assume that all models have the same cost. The resulting meta-model is shown in Figure 5. The meta-model, in this case, is less efficient, asking for a prediction more than once from most of the models, for example, independently of the prediction of model $M0$, the meta-model asks twice for a prediction from $M1$. Also, note that this meta-model decides to use $M1$ instead of $M2$, which is a cheaper model, but not necessarily worse than $M1$. As we can see from Table 5 and the confusion matrix in Figure 6, there is no strong difference between the classification results; only in Cepheids can we see a 2% improvement in the f-score when the meta-classifier does not penalize each model according to their cost, but there is a drop in f-score for the class of Be stars. Calculating the total training cost for the meta-classifier in both cases (with and without considering the cost of the expert models), when the meta-model does not take into account the cost, the training process takes 167% longer than in the case when the meta-classifier takes into account the cost of the model experts.

4.2. EROS Data Set

The EROS project (Expérience de Recherche dObjets Sombres; Derue et al. 1999) observed the Galactic Spiral Arms (GSA), LMC, SMC, and Galactic bulge during 6.7 years, dedicated to detect microlensing events. Observations were done in two nonstandard passbands. One is the EROS-red passband R_E, centered on $\bar{\lambda} = 762$ nm and EROS-visible passband V_E, centered on $\bar{\lambda} = 600$ nm. The light curves used in this work are from the LMC (60 fields) and SMC (10 fields). The limiting magnitude of the EROS V_E band is ~ 20. The cadence varies among the fields, but, in average, about 500 observations were obtained for each light curve. The training set contains 68,718 labeled light curves, obtained from Kim et al. (2014). Table 6 shows the number of light curves per each of the available classes. This training set is more complex than the MACHO training set, in the sense that some subclasses of variability are added to the problem, making the separation more challenging due to the similarity among some classes. Our main goal is not to solve the classification problem for all of the subclasses, but to solve the integration problem using the provided expert models. Therefore, in cases where the respective experts do not classify some subclasses well, the meta-model will probably not be able to classify those classes well either. We used six model experts, each one trained on a specific problem, with a specific set of features. Table 7 shows the features used on each model, the available classes each model can predict and the average cost per light curve that the model takes to perform classification.

Table 8 shows the precision, recall, and f-score of each of the classes per model. As we can see, in some cases, the experts failed to classify some of the classes. For example, $M3$ it not able to successfully classify T2 Cepheids and also the f-score for Cepheids 1O 2O is lower than the average score of the other models and classes. This setup, in particular, shows us that some of the variability classes cannot be automatically classified by the expert, making the meta-model learning process harder than the setup with MACHO data set.

Figure 7 shows the resulting meta-model for the EROS training set. We can see that at the root level, the meta-model asks $M4$ for a classification, in cases where $M4$ predicts LPV SRV AGB C, LPV Mira AGB C, LPV Mira AGB O, and LPV SRV AGB O, the meta-model believes $M4$. In other cases, it asks for other predictions. This makes sense because $M4$ is the only model trained to separate the subclasses of LPVs. In some cases, the meta-model wants to be more confident about the prediction of some of the LPV subclasses, asking other models and predicting the LPV subclasses again when most of the other models predict "Others." When $M4$ predicts LPV OSARG RGB O, the meta-model asks for more information before making a final decision. For example, asking $M0$, and in cases where $M0$ is not so confident about one of its classes and predicts "Others," the meta-model predicts according to $M4$. Another model that contributes extra information about the LPV stars is $M1$, which can predict between "LPV" or "Not-LPV." We can see from the tree that, in some cases, the meta-model ends up predicting a subclass of LPVs after most of the models predict "Others" and $M1$ predicts an LPV. It is interesting to see how the meta-model takes advantage of having more experts trained to classify RR Lyrae stars. For example, after $M4$ predicts "Others," the meta-models ask for $M3$, and when $M3$ predicts an RR Lyrae, instead of immediately believing it, the meta-model asks $M2$, and again if $M2$ predicts an RR Lyrae, the meta-model also predicts RR Lyrae. More interesting is when $M4$ says "Others" and $M3$ also says "Others." If $M2$ predicts an RR Lyrae, the meta-model asks for a prediction from $M5$ instead of immediately believing $M2$, and if $M5$ confirms that it is an RR Lyrae, then the meta-model also predicts an RR Lyrae.

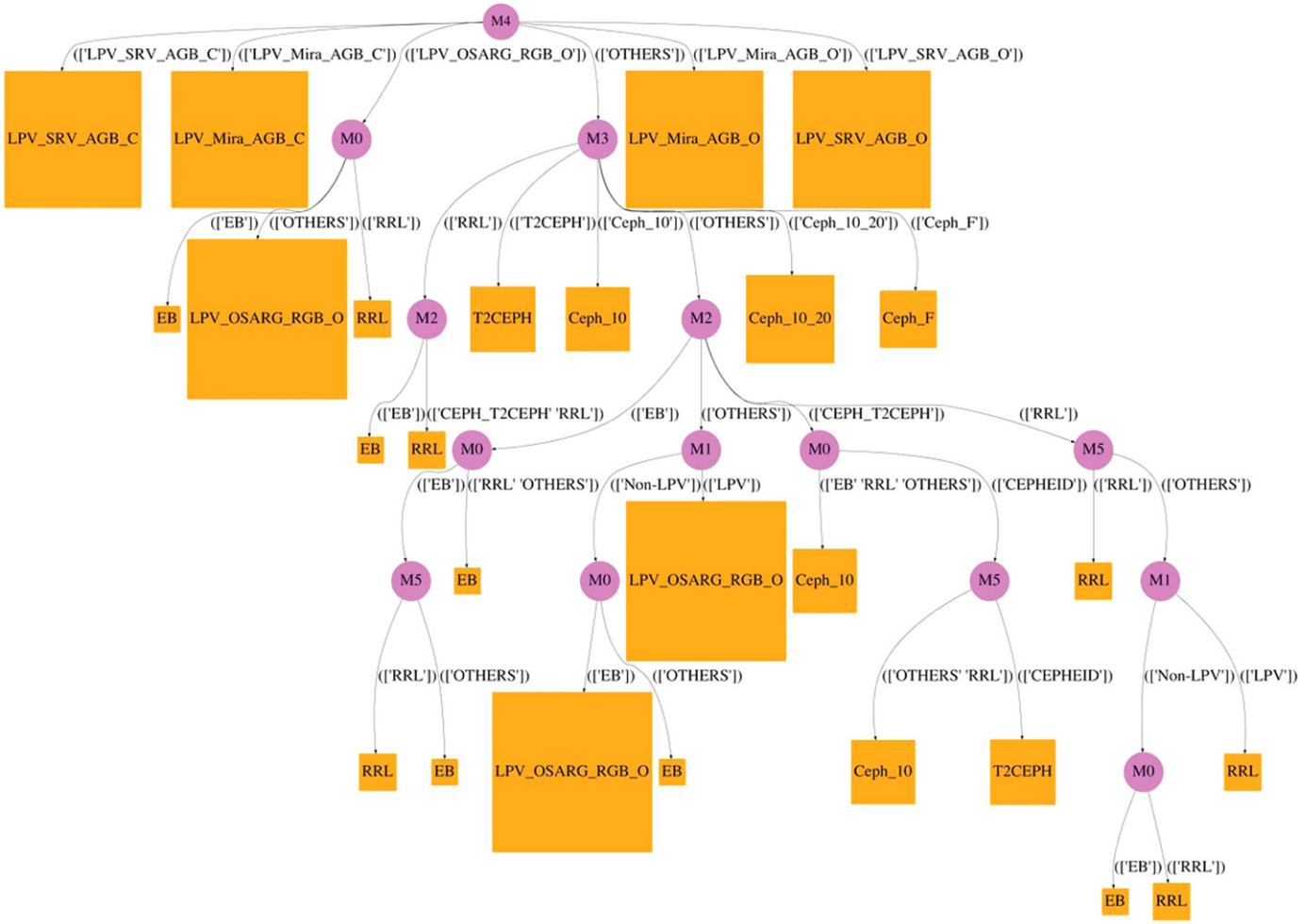

Figure 7. Big picture of the Meta-model learned from the EROS training set.

Table 9
Accuracy Indicators per Class for the Meta-model in the EROS Training Set

Class	Precision	Recall	F-Score
Ceph 1O	0.901	0.878	0.889
Ceph 1O 2O	0.758	0.676	0.714
Ceph F	0.965	0.965	0.965
EB	0.938	0.954	0.946
LPV Mira AGB C	0.872	0.867	0.869
LPV Mira AGB O	0.904	0.794	0.845
LPV OSARG RGB O	0.974	0.990	0.982
LPV SRV AGB C	0.947	0.921	0.934
LPV SRV AGB O	0.901	0.863	0.882
RRL	0.965	0.938	0.951
T2CEPH	0.893	0.545	0.677

To show that the meta-model does not sacrifice performance after the integration, Table 9 shows recall, precision, and f-score of the final meta-model. In Figure 8, we can see the confusion matrix of the meta-classifier. Most of the recall, precision, and f-score values are maintained in the meta-classifier.

As we did in the MACHO experiment, in EROS, we also run the same experiment without considering the cost of each model. The resulting meta-model is shown in Figure 9. Similarly to that in the MACHO case, the meta-model asks many times for a prediction from most of the models, trying to maximize the confidence about the prediction instead of counting how expensive the process is. The meta-model basically asks all of the models that can contribute some information about certain classifications, maximizing the

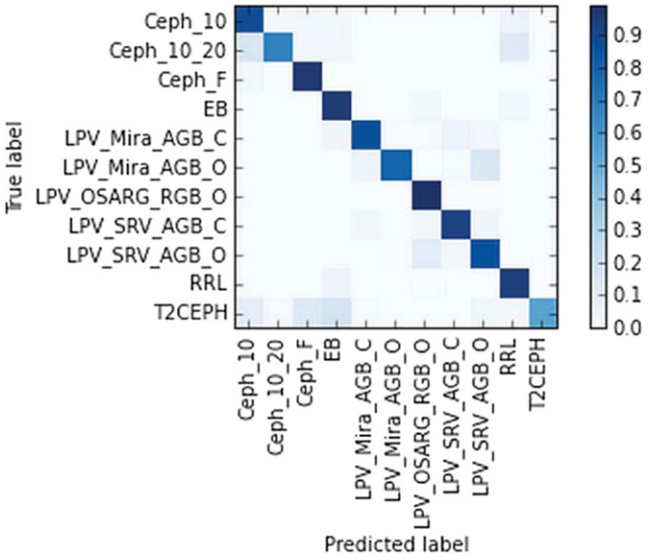

Figure 8. Confusion matrix for the meta-model learned from the EROS training set.

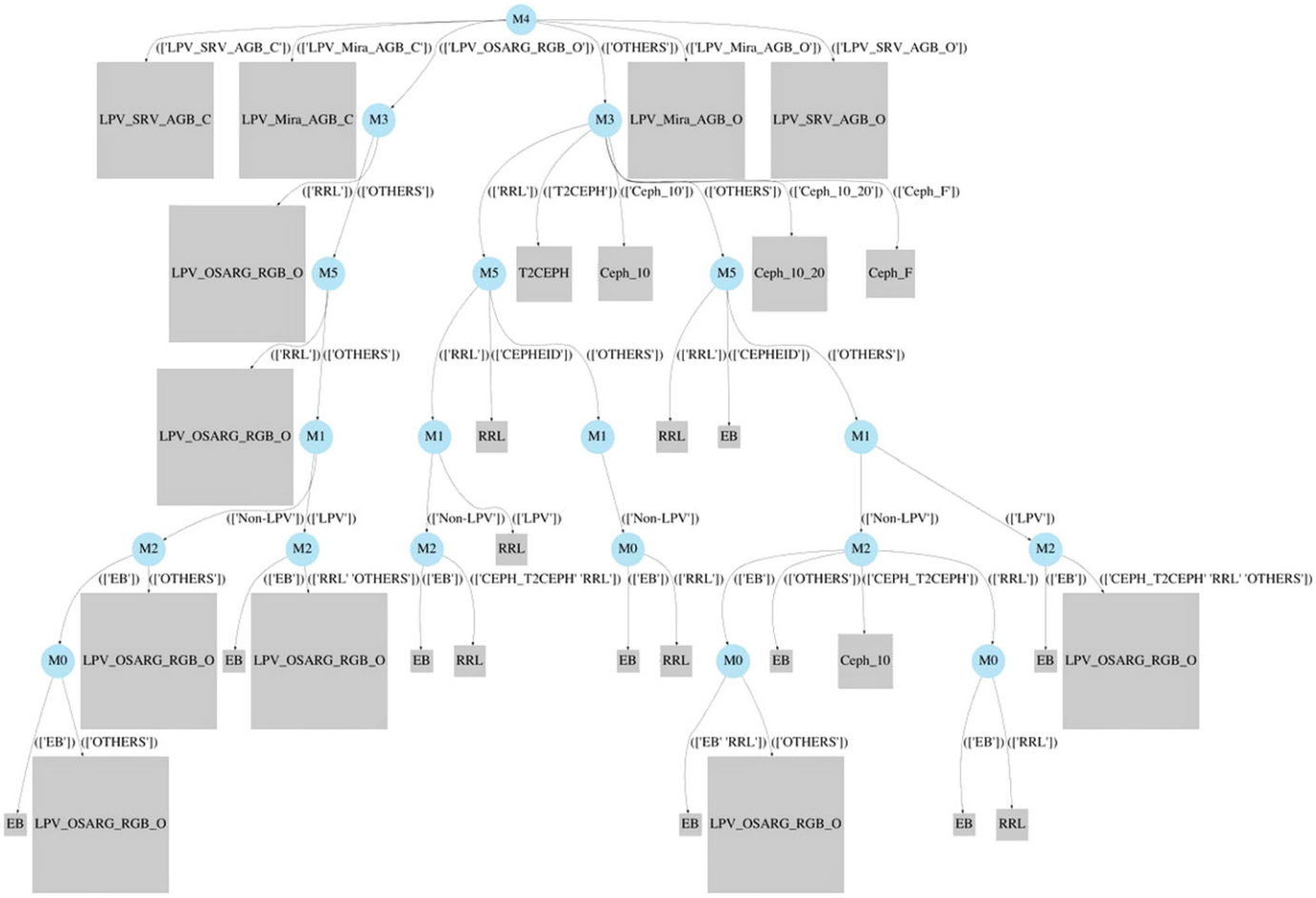

Figure 9. Meta-model learned from the EROS training set without considering the cost of the models.

Table 10
Accuracy Indicators per eClass for the Meta-model without Considering the Cost of Models in the EROS Training Set

Class	Precision	Recall	F-Score
Ceph 1O	0.910	0.876	0.893
Ceph 1O 2O	0.758	0.676	0.714
Ceph F	0.962	0.965	0.963
EB	0.936	0.955	0.945
LPV Mira AGB C	0.872	0.866	0.869
LPV Mira AGB O	0.904	0.794	0.845
LPV OSARG RGB O	0.973	0.990	0.982
LPV SRV AGB C	0.947	0.921	0.934
LPV SRV AGB O	0.901	0.863	0.882
RRL	0.966	0.936	0.951
T2CEPH	0.931	0.545	0.687

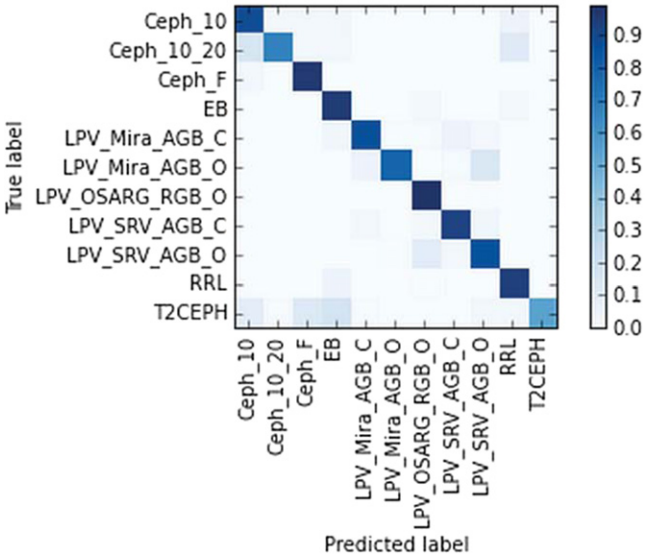

Figure 10. Confusion matrix for the meta-model without considering the cost of the experts, learned from the EROS training set.

confidence without restriction on the number of questions it asks. We can see, for example, that models *M*3 and *M*5 are the most expensive models (Table 7), so the meta-model that takes into account the cost, does not call to *M*3 and *M*5 as much as the meta-model that does not consider cost. From Table 10 and the confusion matrix in Figure 10, we can see that there is no significant improvement in f-score in the meta-model that does not consider the cost. Calculating the total training cost for the meta-classifier in both cases (with and without considering the cost of the expert models), when the meta-model does not take into account the cost, the training process takes about 80% longer than in the case when the meta-classifier takes into account the cost of the model experts.

5. CONCLUSIONS

We present a novel algorithm that allows astronomers to solve new classification problems by reusing previously trained classifiers. These kinds of solutions facilitate a faster development of automated classification methodologies, avoiding the need to retrain new models from scratch. Upcoming surveys such as LSST (Ivezic et al. 2008) will demand this kind of solution since the amount of data will not allow scientists to waste time recalibrating models every time new scientific problems appear. Our intuition is that when a new variable star classification problem arises, if there are classes of stars and features already involved in previous problems, we should be able to use those models in the building process of the new solution. So far, most of the research done in the field of automatic classification of variable stars shows strong relationships among the classes studied and the features used. Any of those classifiers could be plugged into our meta algorithm and be used to build a new solution. An important contribution of this work lies in the possibility of working with different contexts, something that is very natural when model integration occurs; every model has to deal with its own classes and its own data representation, which makes the integration more challenging. So far, we have very promising results. The accuracy of the meta-model was as good as the accuracy of the model experts, which is the first goal that an integration model must achieve.

Another important contribution is that the meta-model is human readable. We can easily observe the meta-model structure, directly inferring how the meta-model acts on every possible situation, making the meta-model more trustable for scientists. In future research, we aim to work on the integration of data coming from different kinds of telescopes. This creates new challenges to overcome, such as the identification of hidden patterns that come from instrumental differences, and the application of those patterns to the classification models to make them able to work on heterogeneous data. We strongly believe that making efforts in that direction will have a huge impact in the astronomical community. An issue that is not addressed in this work is the fact that the training sets are unbalanced and not properly evaluated. Analyzing and generating better training sets is a future research direction. As a matter of fact, there are no good descriptions on how most of the training sets were generated in the first place. For this work, we assume the training sets are given. Fortunately, from the results, we can see that Random Forest classifier can deal with unbalanced training sets. The k-fold cross-validation process we use is stratified, ensuring that the testing and training sets are created with the same proportions of stars as the initial variability classes.

REFERENCES

Bazell, D., & Aha, D. W. 2001, ApJ, 548, 219
Belokurov, V., Evans, N. W., & Du, Y. L. 2003, MNRAS, 341, 1373
Bishop, C. M. 2006, Pattern Recognition and Machine Learning, Vol. 4, URL http://www.library.wisc.edu/selectedtocs/bg0137.pdf
Bishop, C. M., & Svensen, M. 2012, Bayesian Hierarchical Mixtures of Experts, URL http://arxiv.org/abs/1212.2447
Bloom, J. S., & Richards, J. W. 2011, in Advances in Machine Learning and Data Mining for Astronomy, ed. M. J. Way (Abingdon: Taylor & Francis), 89
Bloom, J. S., Richards, J. W., Nugent, P. E., et al. 2011, PASP, 124, 921
Breiman, L. 2001, Machine Learning, 45, 5
Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. 1984, Classification and Regression Trees, Vol. 19 (New York: Chapman & Hall)
Butler, N. R., & Bloom, J. S. 2011, AJ, 141, 93
Chamroukhi, F. 2015, Non-Normal Mixtures of Experts, 61, URL http://arxiv.org/abs/1506.06707
Cook, K. H., Alcock, C., Allsman, R. A., et al. 1995, Variable Stars in the MACHO Collaboration Database, 10 , URL http://arxiv.org/abs/astro-ph/9505124
Derue, F., Afonso, C., Alard, C., & Albert, J.-N. 1999, Observation of Microlensing towards the Galactic Spiral Arms. EROS II 2 year survey, 11, URL http://arxiv.org/abs/astro-ph/9903209
Derue, F., Marquette, J.-B., Lupone, S., et al. 2002, A&A, 389, 149
Faraway, J., Mahabal, A., Sun, J., et al. 2014, Modeling Light Curves for Improved Classification, 16, URL http://arxiv.org/abs/1401.3211
Freund, Y., Schapire, R., & Abe, N. 1999, TJSAI, 14, 5
Gran, F., Minniti, D., Saito, R. K., et al. 2015, A&A, 575, A114
Huijse, P., Estevez, P. A., Protopapas, P., Zegers, P., & Principe, J. C. 2012, ITSP, 60, 5135
Ivezic, Z., Tyson, J. A., Abel, B., et al. 2008, LSST: from Science Drivers to Reference Design and Anticipated Data Products, 39, URL http://arxiv.org/abs/0805.2366
Jordan, M. I., & Jacobs, R. A. 1993, in Proc. International Joint Conf. on Neural Networks, Vol 1, 1339
Jordan, M. I., & Jacobs, R. A. 1994, Neural Computation, 6, 181
Kaiser, N. 2004, Proc. SPIE, 5489, 11
Kelly, B. C., Bechtold, J., & Siemiginowska, A. 2009, ApJ, 698, 895
Kim, D.-W., Protopapas, P., Byun, Y.-I., et al. 2011, ApJ, 735, 68
Kim, D.-W., Protopapas, P., Bailer-Jones, C. A. L., et al. 2014, A&A, 566, A43
Kim, D.-W., Protopapas, P., Trichas, M., et al. 2012, ApJ, 747, 107
Kuncheva, L. I. 2004, ITNN, 8, 3
Kuncheva, L. I. 2007, ITNN, 18, 964
Kuncheva, L. I., & Whitaker, C. J. 2003, Machine Learning, 51, 2
LeBlanc, M., & Tibshirani, R. 1996, J. Am. Stat. Assoc., 91, 1641
Lin, Y.-Y., & Liu, T.-L. 2005, in IEEE Computer Society Conf. on Computer Vision and Pattern Recognition CVPR 2005, 1, Robust Face Detection with Multi-class Boosting, 680
Long, J. P., Karoui, N. E., Rice, J. A., Richards, J. W., & Bloom, J. S. 2012, PASP, 124, 280
Meir, R. 1996, Non-linear Models for Time Series Using Mixtures of Experts
Nun, I., Pichara, K., Protopapas, P., & Kim, D.-W. 2014, ApJ, 793, 23
Nun, I., Protopapas, P., Sim, B., et al. 2015, FATS: Feature Analysis for Time Series, URL http://arxiv.org/abs/1506.00010
Pichara, K., & Protopapas, P. 2013, ApJ, 777, 83
Pichara, K., Protopapas, P., Kim, D., Marquette, J., & Tisserand, P. 2012, MNRAS, 18, 1
Protopapas, P., Huijse, P., Estévez, P. A., et al. 2015, ApJS, 216, 25
Quinlan, J. 1993, C4.5: Programs for Machine Learning (San Mateo, CA: Morgan Kaufmann Publishers)
Quinlan, R. 1986, Machine Learning, 1, 81
Rasmussen, C. E., & Ghahramani, Z. 1991, in Advances in Neural Information Processing Systems, 14, 881
Richards, J. W., Starr, D. L., Butler, N. R., et al. 2011, ApJ, 733, 10
Tisserand, P., Guillou, L. L., Afonso, C., et al. 2007, A&A, 469, 387
Udalski, A., Szymanski, M. K., Soszynski, I., & Poleski, R. 2008, AcA, 58, 69
Wolpert, D. H. 1992, NN, 5, 241
Xu, L., Jordan, M. I., & Hinton, G. E. 1995, in Advances in Neural Information Processing Systems, 633
Zehnder, P., Koller-Meier, E., & Gool, L. J. V. 2008, BMVC, 1