The RAMP framework: from reproducibility to transparency in the design and optimization of scientific workflows

Balázs Kégl & Alexandre Boucaud
LAL, CNRS, Paris-Saclay Center for Data Science
{kegl,aboucaud}@lal.in2p3.fr

Mehdi Cherti & Akın Kazakçı
PSL Research University, CGS-I3 UMR 9217
{akin.kazakci, mehdi.cherti}@mines-paristech.fr

Alexandre Gramfort & Guillaume Lemaitre & Joris Van den Bossche
INRIA, Parietal team, Paris-Saclay Center for Data Science
{alexandre.gramfort, guillaume.lemaitre,joris.van-den-bossche}@inria.fr

Djalel Benbouzid
Data:Lab, Volkswagen Group
djalel.benbouzid@volkswagen.de

Camille Marini
Owkin
camille.marini@owkin.com

Abstract

The RAMP (Rapid Analytics and Model Prototyping) is a software and project management tool developed by the Paris-Saclay Center for Data Science. The original goal was to accelerate the adoption of high-quality data science solutions for domain science problems by running rapid collaborative prototyping sessions. Today it is a full-blown data science project management tool promoting reproducibility, fair and transparent model evaluation, and democratization of data science. We have used the framework for setting up and solving about twenty scientific problems, for organizing scientific sub-communities around these events, and for training novice data scientists.

1 Introduction

As the world's data generation accelerates, machine learning (ML) has become more important than ever for sciences. Scientific data, however, is scattered across communities, sub-communities, or even at the hands of a handful researchers who are, more often than not, disconnected. Today, still too many research projects are being designed in a way to find and recruit, often temporarily, a researcher with some machine learning and data analysis skills for a limited period of time, hoping that significant progress will be made during that time. Under this very common working scheme it might be argued that the bottleneck for scientific progress, at least as far as experimental data-driven science goes, is the lack of top-level, highly-skilled data scientists. In our experience, this view is wrong. In data-driven scientific activities, the essential problem is i) the design of the research project and setup, and ii) the evaluation, in full transparency, of its relevance and significance.

The problem is essentially organizational. Most researchers are either unaware that there is a better way to organize projects, or not incentivized enough to seek such organization. Indeed, the bulk

Preprint. Work in progress.

of the researchers still make their careers by virtue of the publications they produce or marginal improvement they bring to some previous work. While, in theory, any such progress is commendable, this conception of science completely overlooks the fact that a large part of science is done today in fragmented communities and by researchers who are neither willing nor required to provide reproducible work. The real bottleneck in a modern, data-driven experimental scientific paradigm is this inability of the scientific community as a whole to quickly benchmark and assess the scientific worth of a dataset and its analytics setup. Under such perspective, the problem moves from that of reproducibility towards a problem of *transparency* and *organization of work*.

While these issues also apply to core ML research to some extent, the situation is somewhat different here. Through the nineties to the new millennium, ML research was structured naturally around benchmark datasets that allowed comparison (albeit with some reproducibility issues). This situation naturally pushed the ML community towards competition on the quality of algorithms. However, this has been changing. We now have algorithms that perform extremely well on previously established benchmarks. Naturally, the next step is to establish new benchmarks, both inside and outside ML. This has a direct consequence on reproducibility since it implies that what needs to be verified, the scientific result, is no longer only the performance of an algorithm within a given setup, but the *setup itself*.

This requires a major reconsideration of what a scientific result is under a data-driven experimental paradigm and what should be the object of the new scientific competition. Competition on a well-defined metric and a given data set, alone, cannot be counted as science. Does a winning submission on Kaggle constitute a scientific result? The issue is under-determined: what does separate a well-tweaked algorithm from scientific progress? More importantly, what kind of tools and approaches does the worldwide scientific community need *today* to reconsider this question? The issue is nothing less than finding ways to incentivize scientists to contribute in all the various aspects that needs to be addressed, whether it is competition under a well-defined metric or to setting up the task following well-defined, transparent, and shared standards.

The ML community has gone great lengths to standardize predictive *models* and APIs (scikit-learn [1, 2], keras [3], etc.) which yielded major results in terms of model reusability. We made much less effort in standardizing the predictive *experiment*, more generally the *data science process*. ML research is organized in terms of challenges for major problems (e.g., ImageNet [4] for image classification) with large sub-communities agreeing on experiments and setup. This was arguably an important enabler of deep learning as a major breakthrough: the community could verify results easily. But predictive problems come in all shapes and colors, and little effort has been put into generalizing the data challenge paradigm into the tails of predictive problems, that is, subproblems on which only small pockets of researchers work in isolation, often on different data sets, using different metrics and different experimental designs.

Reproducibility (for instance, submitting a docker container with every experimental paper) is certainly useful, but we argue that it is not enough. We need to make sure that experiments are properly set up, the community agrees on the data set, the experimental pipeline, and the metrics used to evaluate the solutions. We need to organizationally separate setting up experiments from working on solutions. Moreover, we need to make sure that formalizing predictive problems and setting up these experiments is as important a research contribution (if not more) as advancing algorithmic solutions on these problems. These principles come from our experience and work around the RAMP framework.

The paper is organized as follows. In Section 2, we describe how RAMP was born and the key problems it addresses, and summarize our design goals. We then present the anatomy of the RAMP kit in Section 3, show some technical meta analysis in Section 4, enumerate the typical use scenarios in Section 5, present several example RAMPs in Section 6, and conclude in Section 7.

2 Development of RAMP: what we learned

RAMP is a software framework that we have started to develop in 2014 for a specific supply/demand unbalance at the Paris-Saclay Center for Data Science. We had several (domain) scientific groups which wanted to apply machine learning on their problems, yet few available data scientists who could or were willing to work with them for an extended time. The idea was to build a lightweight collaborative studio for rapid prototyping and optimization of predictive workflows and run intensive

(typically single-day) datathons with data science researchers. The setup worked well, launching and concluding several interdisciplinary projects.

Collaboration through code sharing was a key: we found that classical competitive data challenges are slow, participants do not exchange their ideas, wasting precious data scientist time. We also realized that *prediction-submission* challenges are useless for the domain scientists since they do not deliver reproducible models and production-ready code. Code submission solved both of these problems with the price of having to develop a heavy cloud management backend to take care of training and testing the submitted models.

Through running half a dozen datathons, we developed some key insights. First, we started to use RAMPs in data science *training*, and since we needed to grade the students, we introduced a closed (competitive) phase. We then realized that they actually worked better (in terms of improving the score). We conducted some controlled experiments and found that full collaboration from the start is suboptimal, experimenters get *fixated* on the initial solution we provided; a well known phenomena in innovation management [5]. From that point on we have been running all challenges as a hybrid: a competitive phase, wherein the participants have access to the model scores only, followed by a collaborative phase in which, additionally, the source code of the models can be unveiled. Second, we found that number beats experience: 80 novice students in three afternoons beat three experienced researchers working for a week or a PhD student or intern working for six months (again, in terms of prediction score) *once the predictive workflow has been set up to lower entry level*. Third, we realized that in terms of time, the bottleneck became *setting up* the experiments. From then on we have been thus streamlining and standardizing the procedure to set up a RAMP. Through several iterations we fine tuned the software: we identified four *moving parts* (the workflow, the scores, the data connectors, and the cross-validation object) to parametrize the RAMP bundle, and wrote a script, unique across RAMPs, that runs the training and testing. The script serves i) for testing the experimental setup by the designer, ii) for testing the submitted models by the experimenter, and iii) for running the training at the backend of the online studio. The main outcome of this modularization was that it reorganized work: it separated setting up the experiments from solving and optimizing the models; a practice which is followed by most of mature sciences that use the *experimental paradigm*.

2.1 Design goals and principles

In this section we enumerate the design principles that we implemented in the RAMP framework. Some of these principles were specifications that existed before we started to develop the framework, some others we learned along the way and then applied consciously, and a third group is good practices that automatically followed from the design but which are nevertheless important to explicitly articulate. We explain in Section 3 how we implemented these principles in detail.

Separating experiment building from optimization ensures that optimization does not start until the experiment is fully specified. We usually organizationally separate *experiment designers* who design the RAMP kit and take care of the data collection and splits, *challenge organizers* who are responsible for funding the infrastructure and possible prizes, communication, and possibly registration, and *experimenters* who participate in the RAMP by submitting models. **Transparency and reproducibility** follow from code submission. Each model needs to be executed on the backend for being scored. We use a two-layered version control (challenge and submission level) and containarize training to assure that model submitters have full control over the (open source) libraries they need for their models. Providing a unique training/testing script (Section 3.3) assures that local training and backend training are programmatically the same (only that they are usually run on different data splits). **Fairness and democratization** are ensured by keeping the test data hidden and by communicating test scores rarely, typically at the end of an official data challenge, and only for a small set of pre-registered models. Our backend is separated from the submission site. It is fully compatible with AWS (Amazon Web Services) and would be easy to extend to other cloud computing platforms, so the framework is scalable in terms of size and number of models, given sufficient funds. We can measure resource use and can set resource limits. We modularize the predictive workflow as much as possible and provide a syntactically functional starting kit (Section 3.2). These lower the entry threshold so novice experimenters can contribute substantially with little technical background. **Making it easy to collaborate.** In the open phase, experimenters have access to each other's models, accelerating model development by collaborating. The first RAMP versions used solely GitHub for this purpose and it is still used as a complementary communication tool, but it was not natural to

rely only on GitHub since i) we need to be able to control the privacy of the test data and of code in the competitive phase, and ii) we need to maintain concurrent models in an "or" relationship which clashed with GitHub's sequential (version-based) design.

3 The RAMP kit

In this section we summarize the main components of the RAMP framework. All of the RAMP kits of the 18 experiments listed at the submission site are available openly at `https://github.com/ramp-kits`. The wiki of the `ramp-workflow` library [6] provides detailed information on the components of the framework. In particular, we explain how to build a RAMP kit bundle here, and explain the anatomy of the RAMP kit here. Some specific RAMP kits are presented in Section 6.

3.1 The bundle

The core design element of the RAMP ecosystem is the **bundle that defines the predictive experiment or problem**. It is encapsulated into a single Python script called `problem.py` (Figure 1). It has four major components.

1. The **workflow** describes the computational elements that the data will go through to generate predictions. The simplest workflows may contain a single workflow element (e.g., a classifier, a regressor, or a detector), but most of the time we modularize the workflow at least into a feature extractor and a predictor to help experimenters to contribute according to their expertise. Each workflow element is defined by (typically) a Python class, following standard signature from scikit-learn as much as possible. The workflow itself is similar to scikit-learn's pipeline but more flexible, especially in defining its interaction with the cross validation component. Once the experiment is set up, experimenters will optimize and submit a single Python file for each workflow element defined in the bundle.
2. The **score types** (or metrics) define how the predictions are evaluated. Experiment designers can define as many scores as needed. Score types typically take a ground truth vector $\mathbf{y}$ and a prediction $\hat{\mathbf{y}}$ output by the last element on the workflow and generate a real-valued score. Most commonly used metrics are available in ramp-workflow, while custom-made metrics can be defined in the bundle.
3. The **cross-validation object** defines how the training and evaluation will be carried out. Most of the time we use off-the-shelf scikit-learn objects, but it is often customized (e.g., in Figure 1 we were asked to check whether the models were robust across experiments/replicates). A special time-series cross-validation object is part of the ramp-workflow library (see Section 6.3).
4. The **data connectors** are custom made functions that read raw data files and return an input object $\mathbf{X}$ and ground truth $\mathbf{y}$ which are indexed by the cross-validation object and passed to the first element of the workflow and to the score types. While $\mathbf{X}$ or $\mathbf{y}$ are Numpy arrays or eventually Pandas dataframes, the input can take any form provided it can be read with Python (e.g., CSV, hdf5, netcdf, SQL database etc.).

A major design goal was to make the setup customizable and extensible. Experiment designers can either use and parameterize off-the-shelf components from the ramp-workflow library or define the component in the bundle. Typically, in the development cycle, new components are first designed in a particular bundle, then, if we foresee that the component will be reused, they are put on the shelf in the ramp-workflow library.

3.2 The starting kit

The starting kit is a simple, **computationally lean example submission** that satisfies the formal requirements of the workflow defined in the bundle. It has a double role. First, it lowers the entry threshold for experimenters by allowing them to start with a syntactically valid basic solution instead of a white sheet. Second, it serves to the experiment designer for the unit testing of the bundle itself. Besides debugging the bundle, it also helps the experiment designer to set up the experiment (especially the cross-validation object) properly.

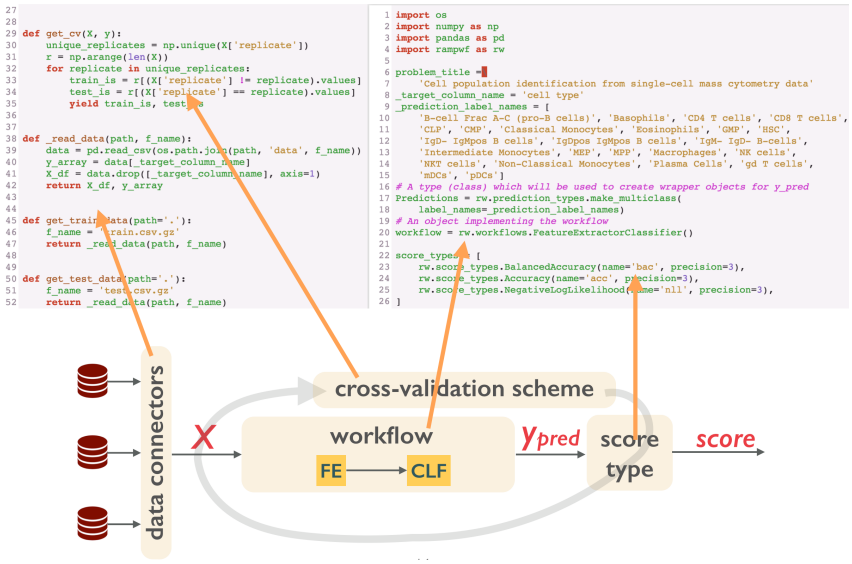


Figure 1: An example script (bundle) parameterizing an experiment. The designer can customize the workflow (computational steps), the score types (metrics), the prediction type, the data connectors, and the cross validation scheme. Frequently used components are available in the ramp-workflow library [6], making the bundle concise, but all elements are fully customizable if needed (in this case: the cross-validation scheme).

3.3 The `ramp_test_submission` script

Once the bundle and the starting kit are defined and the data is in place, a single script called `ramp_test_submission` can be used to run the experiment. It is a unique script that reads all its parameters from the bundle `problem.py` so it requires no modification when setting up a new RAMP kit. This separation of *moving parts* of the experiments into `problem.by` and *fixed parts* into `ramp_test_submission` was a major design breakthrough towards the standardization of the (predictive) data science process.

The script has several roles:

1. It allows experimenters to unit test and rapidly cycle through their experimental optimization loop.
2. It allows the experiment designers to unit test the experiment itself and to tune the bundle.
3. It allows the challenge organizers to outsource the development of a new experiment by verifying compliance rapidly.
4. It allows the challenge organizers to run the challenge backend and to assure that the experimental design is the same in the RAMP kit and at the challenge backend.

3.4 The notebook

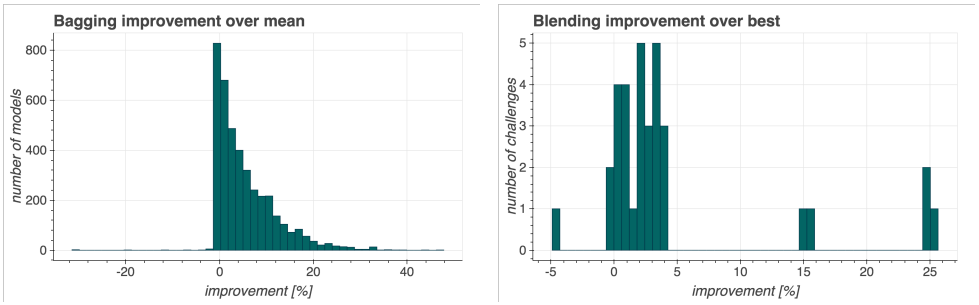
The Jupyter notebook in the ramp kit is not required for running the workflow, but it is strongly advised if the ramp kit is designed for collaborative work. Its role is to **incentivize data scientists to work on the problem** by explaining them the scientific goal, and to make them efficient by providing enough exploratory and explanatory analysis so they understand the data, the predictive goal, and the workflow. The notebook is mandatory for all challenges uploaded to the ramp.studio site.

4 Technical innovations and meta analysis

Besides standardizing and structuring predictive experiments, we have also made a couple of contributions to model evaluation and combination. First, instead of taking the mean cross validation

scores, we bag the models and **score the bagged model**. This technique, to which we refer to as "CV bagging", consistently improves the scores (Figure 2(a)), yet it is rarely used in machine learning research papers. Second, we automatically **combine the models** using the greedy feedforward model selection procedure of [7]. We found this simple but ingenious technique so robust and resistant to overfitting (Figure 2(b)) that we could apply it on the validation data, which means that we did not need to complicate our double validation setup into triple validation, greatly simplifying the experimental setup. In addition, we are running the blending algorithm on each CV fold separately which means that we can compute a graded "contributivity" score of each submitted model. This score, besides model quality, also favors model diversity, giving an important feedback to participants whether their solution is significantly different from the bulk of the submissions.

These analyses are only two simple examples of the meta analyses that we are carrying out using the growing meta data on the RAMP site.



(a) (b)

Figure 2: (a) The percentage improvement of CV-bagged test scores over the mean CV test scores over 4000 models submitted to 26 challenges on 12 problems. The median improvement is 3.4%. (b) The percentage improvement of the test score of the blended model over the test score of the best model (best according to validation score) over 33 challenges on 18 problems. The median improvement is 2.4%.

5 Typical use scenarios

Although the main goal of the framework is to feed the online studio with standardized experiments, the framework can also be used by **single users** or **small teams** for organizing experimental research. Obviously, the test set cannot be hidden in this case so proper experimentation will require self discipline. Nevertheless, the separation of experiment building and experimentation and the organization and logging all tested models can improve the efficiency of research. In this "mode", users only need to understand and use the ramp-workflow library [6], build a RAMP kit (Section 3), and organize collaboration (model exchange) through GitHub or other code sharing platforms.

We have run more than a dozen single-day **datathons**, some of them co-organized with research workshops (e.g., the Climate Informatics workshop, the MASSCASUAL workshop, and the Connecting the dots workshop). The goal of these events is **rapid prototyping of predictive solutions**, **inciting research communities to properly formalize their predictive problems and to collect data**, and uniting them around these well-formed predictive problems. In terms of improving the predictive score, data challenges and data camps usually beat single-day datathons, so when performance is important, we usually use datathons to debug and optimize the experiments, making sure that the research subcommunity agrees on the setup, then we organize larger-scale events with non-domain-expert data scientists or students incited to work on the predictive problem.

Chronologically, the second major scenario is when we use RAMPs in **data science training**. We had more than a dozen runs in M.Sc. and late B.Sc. data science training (e.g., 1, 2, 3, 4), in summer schools (e.g., CIFAR'18), and in continuing education. The RAMP usually accompanies theoretical courses or practical data science projects. Students start working (competing) on their own, then collaborate and learn from each other when we open the code. Top students then present their solutions to the class.

We have recently started to run money-prize **data science challenges** (the one on autism classification is open at this time). The main technical challenge was to solve transparent backend resource control for fair evaluation; this has been done using standalone Amazon Web Services (AWS) servers for training and evaluation. In terms of performance, these RAMPs (data camps and data challenges) have been more successful than datathons for essentially three reasons: i) larger number of participants, ii) experimenters are incited more (grades or money is at stake), and iii) the hybrid open/close setup.

The RAMP is technically ready to **accompany core ML research**. Today, a typical ML research paper is responsible for data collection, experimental setup, funding backend, running the experimental optimization (testing ideas), and evaluation. This could be changed into a two-tier organization whereas a committee (similarly to a workshop organizing committee) is responsible for the experimental setup, the data collection, the evaluation, and even funding the backend, whereas research groups would only carry out the algorithmic development. One important obstacle at this time is that setting up proper experiments is not valued as research, especially compared to testing and publishing new ideas, whereas in terms of accumulating collective knowledge this activity is arguably as important as algorithmic development.

6 Examples

We present six example RAMPs here out of the eighteen (non-test) RAMP problems (experiments) and about thirty events listed at the RAMP site. We have about 1700 registered users who submitted about 8000 predictive models to these RAMPs.

6.1 Anomaly detection at the LHC ATLAS detector [8]

The goal of the RAMP was to **detect simulated anomalies** in the LHC ATLAS detector. It was a straightforward binary classification problem on sixteen numerical features but it was challenging because of the **complex features needed to be "engineered" for improving the score** (Figure 3). None of the machine learning techniques (random forests, deep learning, etc.) worked well out of the box (even when using global optimization tools like hyperopt). About 10% of the $\sim$100 novice master students improved the score by a large margin in the competitive phase by reading the notebook and understanding the features and the physics. Since their code was closed at this time, they developed diverse solutions on which our blending algorithm worked very well, opening a "diversity gap" between the best model and the blended model. Opening the leaderboard had two major consequences: the rest of the students read and understood these solutions and caught up rapidly with the leaders, and, more interestingly, top students did a better job at "manually" blending solutions than our automatic blending algorithm. We observe these phenomena regularly on most of our challenges, to a more or lesser extent, depending on the problem and the data.

6.2 Drug classification and concentration estimation from Raman spectra [9]

The goal of this RAMP was to automate the **verification of chemotherapy drugs before their administration**. The semi-industrial pipeline consisted of (non-destructively) collecting Raman spectra from solutions packaged in bags, diffusers, or syringes, then verify the identity of the medication and measure its dosage. The RAMP had a triple significance: i) we **solved a real-life problem** with results that allows the industrialization of the predictor, ii) we custom developed our **most complex predictive workflow** that required to combine classification and regression (Figure 4(a)), and iii) this was the first RAMP which **resulted in a (domain) scientific paper** [10]. We have run this RAMP in four different M.Sc. data camps, in one of which about 80 novice students reached near optimum in just three afternoons (Figure 4(d)).

6.3 Spatio-temporal El Niño forecasting six months ahead [11]

We have been collaborating with the Climate Informatics workshop at NCAR since 2015. In this first RAMP the goal was to forecast the **temperature in the ENSO 3.4 region six months ahead**. El Niño (La Niña) is a phenomenon in the equatorial Pacific Ocean characterized by a five consecutive 3-month running mean of sea surface temperature (SST) anomalies in the Niño 3.4 region that is above (below) the threshold of $+0.5°C$ ($-0.5°C$) (Figure 4(b)). This RAMP featured the first **non-trivial workflow** and cross-validation technique that allows the use of nonparametric predictors in **time**

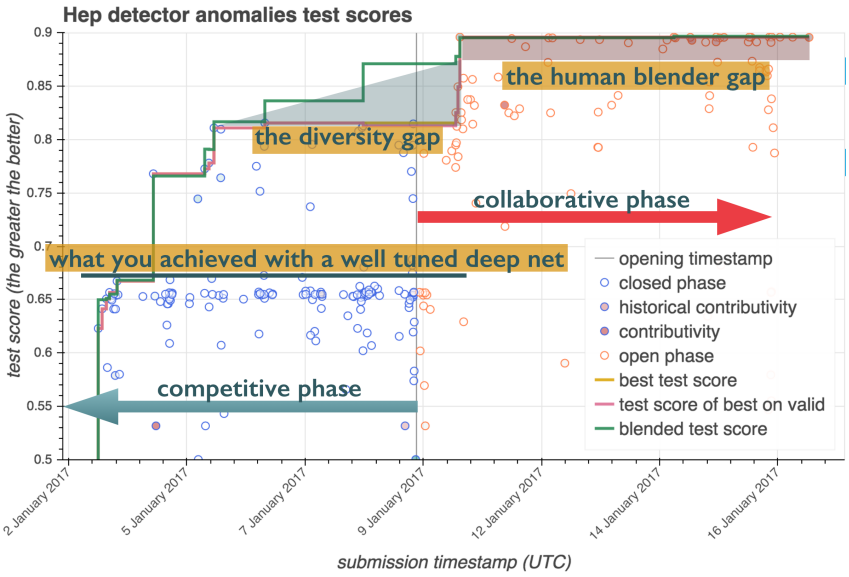


Figure 3: The learning curves of "graduate student descent". A hundred novice students participated in this RAMP for about two weeks. Each circle represents a submission, blue in the competitive phase, orange in the collaborative phase. The score is ROC AUC. The yellow curve is the current best test score, the pink curve is the test score of the current best submission according to its validation score, and the green curve is the test score of blended model.

series forecasting, with a randomized test for checking submissions that illegally use information from the future.

6.4 Image classification of pollinating insects [12]

The goal of this RAMP was to **classify images of insects** from the SPIPOLL crowdsourcing project of the Paris Museum of Natural History, with the scientific goal of **quantitatively studying pollinating insects in France**. In the last edition we had 70K training images and 403 classes with the bottom 100 classes having less than ten images per class, and achieved about 90% accuracy of top class prediction. This was the first RAMP using heavy deep learning models (typically transfer learning pretrained Keras and PyTorch models with $\sim$10h average training time on AWS GPUs), the first offering significant money prizes, and the first RAMP where we **put the classifier into the production**, serving as the backend of this insect classification android app.

6.5 Image detection of Mars craters [13]

The goal of this RAMP was to **detect craters in images taken by Mars satellites**. We have preprocessed the original gigapixel images, cropped them to make them digestible by modern image detection softwares and to handle craters at the edges of the cropped images properly, and implemented a detection workflow and a set of commonly used and new evaluation scores. After a preliminary run this winter, the RAMP will be used to challenge the students of the annual CIFAR Deep Learning summer school 2018.

6.6 Particle tracking [14]

The goal of this RAMP was to **reconstruct particle tracks** in a simulated high-energy particle physics pixel detector. The significance of this RAMP is that it implements a challenge which is **not a prediction problem per se**. We formalized it as a "supervised" clustering task (where cluster identities are known at training time), but the main question here is whether "training" makes any sense or the hand-crafted tracking algorithms are unbeatable. The fact that we could use RAMP for essentially a coding challenge with a quantifiable performance metrics demonstrates its flexibility.

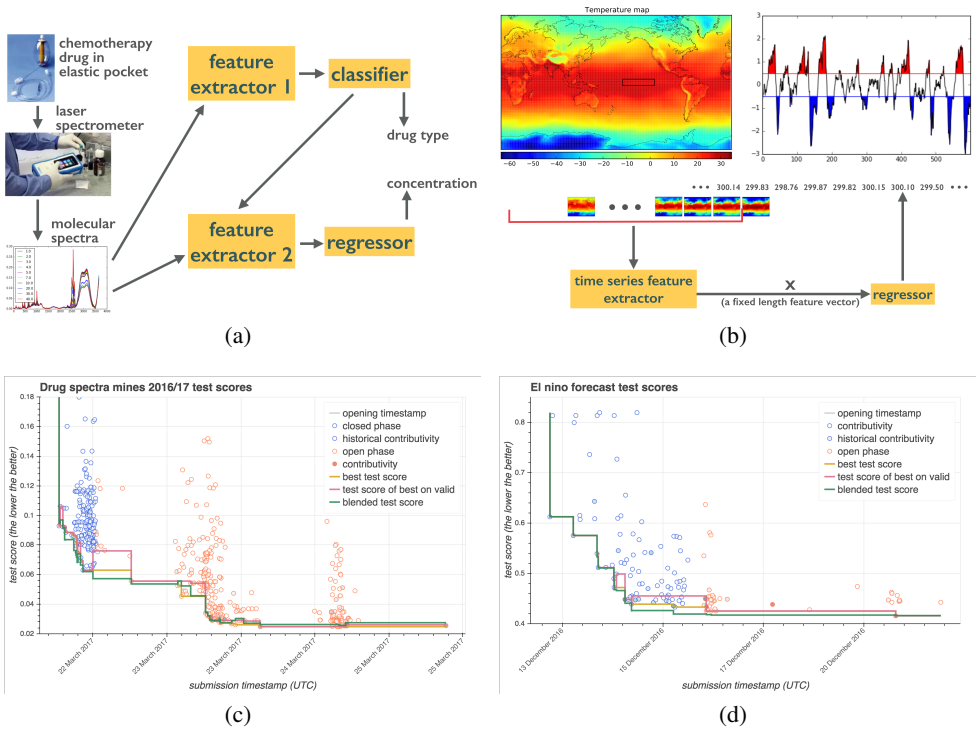


Figure 4: Two nontrivial predictive workflows and corresponding test scores. (a) A combined classification/regression problem. The score is a combination of classification error and concentration prediction mean absolute relative error (MARE). (b) A time series workflow for spatio-temporal forecasting. The score is root mean square error (RMSE) on a single El Niño temperature target predicted six months ahead.

7 Conclusion

We have been using the RAMP framework for organizing work around reproducible data science workflows for scientific projects. We have set up and solved about twenty scientific problems, organized scientific sub-communities around these events, and trained novice data scientists. The framework has been originally designed to accelerate domain science/data science collaborative projects; today it is ready to be used in core ML research for generalizing the data challenge paradigm from the few major experiments where it has been used successfully to the bulk of machine learning problems.

References

[1] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12: 2825–2830, November 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2078195.

[2] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. Vanderplas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. *ArXiv e-prints*, September 2013.

[3] François Chollet et al. Keras. https://keras.io, 2015.

[4] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009. doi: 10.1109/CVPR.2009.5206848.

[5] Marine Agogué, Akin Kazakçi, Armand Hatchuel, Pascal Masson, Benoit Weil, Nicolas Poirel, and Mathieu Cassotti. The impact of type of examples on originality: Explaining fixation and stimulation effects. *The Journal of Creative Behavior*, 48(1):1–12, 2014.

[6] B. Kégl, A. Boucaud, M. Cherti, A. Gramfort, G. Lemaitre, and J. Van den Bossche. The `ramp-workflow` toolkit for building predictive workflows end experiments. https://github.com/paris-saclay-cds/ramp-workflow, 2018.

[7] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble Selection from Libraries of Models. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 18–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi: 10.1145/1015330.1015432.

[8] James Catmore, Imad Chaabane, Sergei Gleyzer, Cécile Germain, Isabelle Guyon, Victor Estrade, Balázs Kégl, Edouard Leicht, Gilles Louppe, David Rousseau, and Jean-Roch Vlimant. The HEP detector anomalies RAMP kit. https://github.com/ramp-kits/HEP_detector_anomalies, 2016.

[9] Camille Marini, Laetitia Minh Maï Le, Alexandre Gramfort, and Balázs Kégl. The drug spectra RAMP kit. https://github.com/ramp-kits/drug_spectra, 2015.

[10] Laetitia Minh Maï Le, Balázs Kégl, Alexandre Gramfort, Camille Marini, David Nguyen, Mehdi Cherti, Sana Tfaili, Ali Tfayli, Arlette Baillet-Guffroy, Patrice Prognon, Pierre Chaminade, and Eric Caudron. Optimization of classification and regression analysis of four monoclonal antibodies from Raman spectra using collaborative machine learning approach. *Talanta*, 184: 260 – 265, 2018. ISSN 0039-9140. doi: https://doi.org/10.1016/j.talanta.2018.02.109. URL http://www.sciencedirect.com/science/article/pii/S0039914018302273.

[11] Balázs Kégl, Claire Monteleoni, Mahesh Mohan, Timothy DelSole, Kathleen Pegion, Julie Leloup, Alex Gramfort, Mehdi Cherti, and Camille Marini. The El Niño RAMP kit. https://github.com/ramp-kits/el_nino, 2015.

[12] Mehdi Cherti, Romain Julliard, Gregoire Lois, and Balázs Kégl. The pollenating insects RAMP kit. https://github.com/ramp-kits/pollenating_insects_3_simplified, 2017.

[13] Alexandre Boucaud, Joris van den Bossche, Balázs Kégl, Frédéric Schmidt, and Anthony Lagain. The Mars craters RAMP kit. https://github.com/ramp-kits/mars_craters, 2017.

[14] Thomas Boser, Isabelle Guyon, Mikhail Hushchyn, Balázs Kégl, David Rousseau, and Yetkin Yılmaz. The particle tracking RAMP kit. https://github.com/ramp-kits/HEP_tracking, 2017.