



HAL
open science

Integrating and reporting full multi-view supervised learning experiments using SuMMIT

Baptiste Bauvin, Jacques Corbeil, Dominique Benielli, Sokol Koço, Cecile Capponi

► **To cite this version:**

Baptiste Bauvin, Jacques Corbeil, Dominique Benielli, Sokol Koço, Cecile Capponi. Integrating and reporting full multi-view supervised learning experiments using SuMMIT. 2022. hal-03845435

HAL Id: hal-03845435

<https://hal.science/hal-03845435v1>

Submitted on 9 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Integrating and reporting full multi-view supervised learning experiments using SuMMIT

Baptiste Bauvin
Jacques Corbeil
*2705, boulevard Laurier,
Québec, Québec,
Canada G1V 4G2*

BAPTISTE.BAUVIN.1@ULAAVAL.CA
JACQUES.CORBEIL@CRCHUDEQUEBEC.ULAAVAL.CA

Dominique Benielli
Sokol Koço
Cecile Capponi
*52 Av. Escadrille Normandie Niemen,
13397 Marseille Cedex 20,
France*

DOMINIQUE.BENIELLI@UNIV-AMU.FR
SOKOL.KOCO@EMSE.FR
CECILE.CAPIONI@LIS-LAB.FR

Editor: Nuno Moniz, Paula Branco, Luís Torgo, Nathalie Japkowicz, Michał Woźniak and Shuo Wang.

Abstract

SuMMIT (**S**upervised **M**ulti **M**odal **I**ntegration **T**ool) is a software offering many functionalities for running, tuning, and analyzing experiments of supervised classification tasks specifically designed for multi-view data sets. SuMMIT is part of a platform ¹ that aggregates multiple tools to deal with multiview datasets such as `scikit-multimodallearn` (Benielli et al., 2021) or MAGE (Bauvin et al., 2021). This paper presents use cases of SuMMIT, including hyper-parameters optimization, demonstrating the usefulness of such a platform for dealing with the complexity of multi-view benchmarking on an imbalanced dataset. SuMMIT is powered by Python3 and based on `scikit-learn`, making it easy to use and extend by plugging one’s own specific algorithms, score functions or adding new features². By using continuous integration, we encourage collaborative development.

Keywords: Multimodal, Supervised, Classification, Benchmarking, Python, Reproducible Research, Modularity, Explainability, Interpretability

1. Introduction

The presented software deals with learning multiple multi-view supervised classification models from a dataset $S = \{(x_i, y_i)\}_{i=1}^n$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, i.i.d. from an unknown distribution D over $\mathcal{X} \times \mathcal{Y}$. In Python’s world, there exists many valuable libraries for handling experimental studies in supervised classification, such as `scikit-learn` (Pedregosa et al., 2011) on mono-view datasets, where each sample x must range in one description space $x \in \mathcal{X}$. These libraries come with tools for performing a wide range of model selection through dedicated pipelines.

1. <https://github.com/multi-learn/>

2. The full documentation is available at <https://multi-learn.github.io/summit/>

In the multi-view setting, \mathcal{X} is actually divided in $V \geq 2$ description spaces (or views; modalities): $\mathcal{X} = \mathcal{X}^{(1)} \times \dots \times \mathcal{X}^{(v)}$. Simply merging the views (*early fusion*) may in certain cases be used to deal with multi-view learning problems. On the opposite, one may perform *late fusion* (Snoek et al., 2005) by learning one model $h^{(v)}$ from each view v , then combining them, usually through majority vote. In contrast to the fusion approaches, the last decades have produced multi-view supervised learning algorithms for taking advantages of diversity and complementarity among views, such as kernel-based MVML (Huusari et al., 2018); boosting-based MUMBO (Koço and Capponi, 2011), or tensor-based, such as Multi-View Machines (Cao et al., 2016), and many more reviewed by (Baltrušaitis et al., 2018). While there exists unsupervised multi-view libraries (Perry et al., 2021), to the best of our knowledge no multi-view supervised classification benchmarking tool is available.

Interestingly, the no free lunch theorem implies that choosing the right model with the right hyper-parameters for a learning task is crucial to obtain relevant results. It is all the more critical in a multi-view context because its nature drastically increases the combinatorics, compared to the mono-view setting. Indeed, some tasks require specific multi-view algorithms, while others are well handled by either naive early or late fusion, complexified by the hyper-parameter optimization in every configuration. Moreover, in some extreme cases, including all the views may actually lower the performance of the models. Indeed, multi-view learning on imbalanced tasks induces more complexity as the information is not uniformly distributed among the views and the classes.

To solve this problem, we propose SuMMIT as an integration tool that allows the users to fine tune and run many mono- and multi-view models at once, together with first-level hyper-parameter optimization, and visual comparison of user-selected performance measures. Moreover, if the desired model is not available in the built-in pool, SuMMIT has been developed to ease the addition of plugins and new algorithms to its workflow, as it is designed to be compatible with other open source multi-view projects such as `scikit-multimodal-learn` (Benielli et al., 2021) and MAGE (Bauvin et al., 2021).

This paper focuses on the main functionalities of SuMMIT by covering its basic usage and some complex features. Technical aspects of SuMMIT’s implementation are presented. A perspective of promising potential work and collaborative development then is given.

2. A multi-view approach to supervised learning

In this section, we introduce the basic definitions and concepts of multi-view learning in a classification context. As an introductory example, let us consider the task of classifying different species of birds, for which the available descriptions are (1) **images**, (2) **audio** recording of several individuals, and (3) **textual** annotation of the place where these information were acquired. Naturally, each description represents a different *view* on the data. In order to understand each view, one has to choose the right model to extract as much information as possible from each data type, and then combine them to build a classifier that takes into account the strengths and weaknesses of the views, and ultimately integrates them all. Thus, to differentiate two bird species that are endemic to the same area but look and sound different, the textual view will be confusing, while the other two should be useful. Similarly, we can suppose that combining as-is a pixel matrix and a bag-of-words representation might confuse a mono-view algorithm, and will not lead to acceptable results.

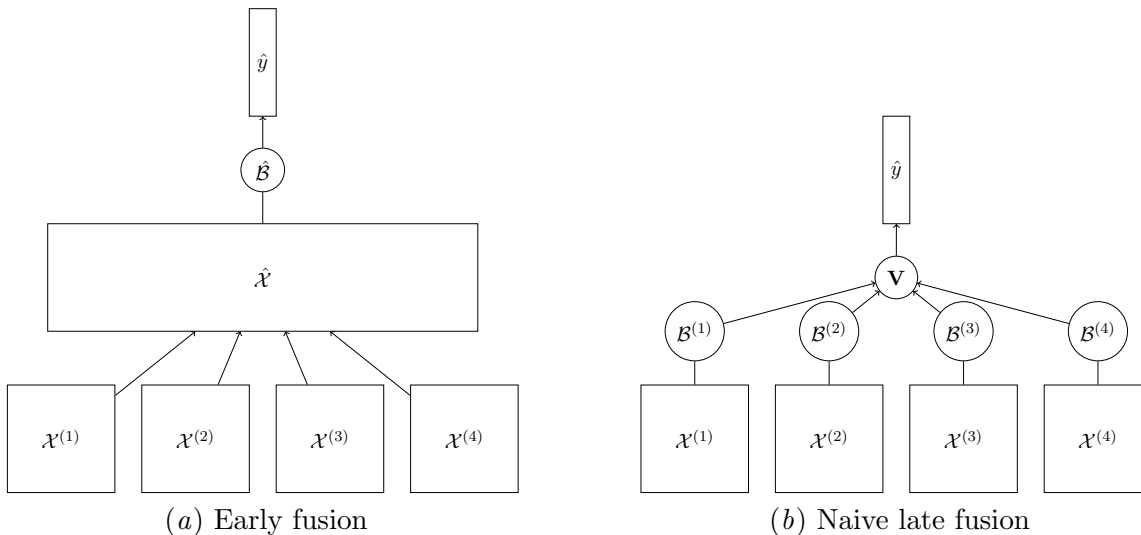


Figure 1: Basic multi-view methods that adapt mono-view approaches.

This type of examples demonstrates the complexity of multi-view learning and provides an insight into the importance of using multi-view specific tools to deal with multi-view tasks.

2.1. Defining a multi-view dataset

In this work, we focus on the supervised classification framework, in which the dataset \mathcal{S} is made of samples \tilde{x}_i and their associated labels y_i , i.e. $\mathcal{S} = \{(\tilde{x}_i, y_i)_{i=1}^m | \tilde{x}_i \in \tilde{\mathcal{X}} \text{ and } y_i \in \mathcal{Y}\}$. In our case, the labels are positive integers, as we work with potentially multi-class datasets $y_i \in \{1, \dots, k, \dots, K\}$. Moreover, in multi-view context, we consider that each sample \tilde{x}_i has been observed thanks to $V \geq 2$ different methods (projections, sensors) and that therefore, we have V observation, or views of $\tilde{x}_i = \{x_i^{(1)}, \dots, x_i^{(v)}, \dots, x_i^{(V)}\}$ with $x_i^{(v)} \in \mathcal{X}^{(v)}$ and $\tilde{\mathcal{X}} = \mathcal{X}^{(1)} \times \dots \times \mathcal{X}^{(V)}$. All these views have their own dimensions $d^{(1)}, \dots, d^{(v)}, \dots, d^{(V)}$. Finally, the concatenation of all the descriptions of a sample \tilde{x}_i is of dimension $D = \sum_{v=1}^V d^{(v)}$.

Armed with these basic notations and definition, we can outline the main approaches used to solve a multi-view problem.

2.2. Solving a multi-view problem

This task adds complexity to the usual machine learning problems. Indeed, the separation among the views can be critical to the model, as shown in the birds example.

Mono-view approaches Even with a multi-view dataset, it is still possible to use basic mono-view algorithms to try to solve the problem. To do so, the the reader has to pick a view and learn a mono-view model on it, ignoring the other $V - 1$ views.

This method reuses the knowledge about mono-view classification as is, without any more investment. Moreover, it can be very computationally interesting as it only requires to analyze one view. However, it is a very high level approach, as it considers that a single view has sufficient information to solve the problem, which is not always the case in real

life multi-view datasets. Unlike what one may expect, this approach sometimes succeeds. The first step towards multi-view approaches is by fusing the views (Snoek et al., 2005).

Early fusion Here, we present the early fusion method that consists in concatenating all the views to create a new set $\hat{\mathcal{X}} \subset \mathbb{R}^D$ that allows any mono-view algorithm to learn on $\hat{x}_i = \left\{ x_{i,1}^{(1)}, \dots, x_{i,d^{(1)}}^{(1)}, \dots, x_{i,1}^{(V)}, \dots, x_{i,d^{(V)}}^{(V)} \right\} \in \mathbb{R}^D$. The process is schematized in Figure 1(a); note that some data processing may need to be performed before and after the fusion.

Empirically, it is possible to use well-studied conventional mono-view algorithms on $\hat{\mathcal{X}}$, so they have access to all the available information about the samples at the same time. However, this new dataset can be of very high dimension if the views are large and/or numerous. In addition, this method does not allow to learn on heterogeneous data types, and even if we supposed here that all the views are real-valued, this can lead to learning difficulties, such as the ones highlighted by the previous bird example. Moreover, when the dimensions of some views are much greater than dimensions of some others, these latter might get neglected by algorithms such as L2-SVM that learn on the full description of each sample. Finally, regularity patterns in the data might be different from one view to another, so searching for regularities in the fused space could be a challenge.

Naive late fusion To adapt mono-view models to a multi-view task, it is also possible to use the late fusion approach (Snoek et al., 2005) that learns one mono-view classifier $\mathcal{B}^{(v)}$ for each view then fuses their decisions in a naive majority vote \mathcal{V} to output a multi-view classification vote $\mathcal{V} = \mathbf{V}_{v=1}^V(\mathcal{B}^{(v)})$ with \mathbf{V} being the majority voting operator. The process is schematized in Figure 1(b).

Empirically, the late fusion method has dual advantages to the early fusion as it allows to process each view at a time (or in parallel) to reduce dimensionality and focus on each view’s regularities. However, it does not benefit from the interactions among the features of every view, nor even able to tackle problems requiring accounting for complementary views.

Native multi-view algorithms The three methods that we introduced are based on mono-view learning, and might not be sufficient to tackle problems that require a larger amount of interactions among the views.

One of the approaches that does include interactions is Mumbo (Koço and Capponi, 2011), a boosting-based algorithm which assumes that views are complementary, therefore that each sample of the dataset is well described in at least one view, but that one view is not sufficient to build a relevant model. The principle of Mumbo is to overweight the relevant views in the classification process of a specific sample thanks to a cost tensor that generalizes the cost matrix of multi-class mono-view boosting.

The authors of Multi-View Machines (Cao et al., 2016) made different hypotheses and supposed that some samples can only be classified when considering multiple-view interactions. Therefore, MVM computes an interaction tensor gathering interactions between an arbitrary number of views and learns a model based on a gradient descent that optimizes a classification loss based on the weights given to each coordinate of the factorization of the interaction tensor. Both these approaches are relevant for specific fields of multi-view machine learning: on one hand, Mumbo might be relevant to learn on datasets where views disagree on the samples, but one has the right description. On the other hand, Multi-View

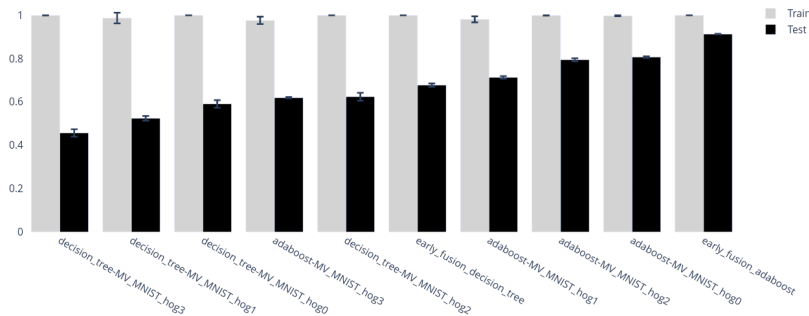


Figure 2: A static version of the interactive accuracy scores bar plot for Adaboost, Decision Tree (on each view and with their early fusion versions), for solving the MV_MNIST multi-class learning task over four views.

Machines might intuitively be relevant for tasks where the collaboration among the views is mandatory to be able to build a relevant model.

The need for a benchmarking tool All the previous models come with their hypotheses, areas of expertise, and are relevant only for specific tasks. Therefore, when working with a specific multi-view dataset, a python tool that allows to evaluate the relevance of groups of models while providing results interpretability or explainability is highly useful.

2.3. The technical difficulties of multi-view learning

Python provides a large amount of machine learning libraries, from `scikit-learn` (Pedregosa et al., 2011) for well-established methods, to `pytorch` for building custom neural networks that continuously push the limits of the field.

However, to the best of our knowledge, no supervised multi-view learning library is available. Indeed, the dataset division in views requires to work with specialized tools that account for the specificities of the context, while proposing already implemented approaches. This is our goal with `multi-learn`³, a multi-view Github organization that regroups `scikit-multimodalllearn`⁴ (Benielli et al., 2021), a library that implements multi-view classifiers based on the `scikit-learn` framework, `MAGE`⁵ (Bauvin et al., 2021), a multi-view dataset generator, and the focus of this paper : `SuMMIT`⁶, a multi-view benchmarking tool. When developing `SuMMIT`, we strived to provide a tool that allows to compare mono- and multi-view approaches on a supervised multi-view dataset, while yielding the maximum amount of information when analyzing the results of the benchmark.

3. Discovering SuMMIT Functionalities

For illustration purpose, we consider the well-studied MNIST (Deng, 2012) supervised classification task. We used our own process to transform it into a four-views dataset. Each

3. <https://github.com/multi-learn>

4. <https://github.com/multi-learn/scikit-multimodalllearn>

5. <https://github.com/multi-learn/mage>

6. <https://github.com/multi-learn/summit>

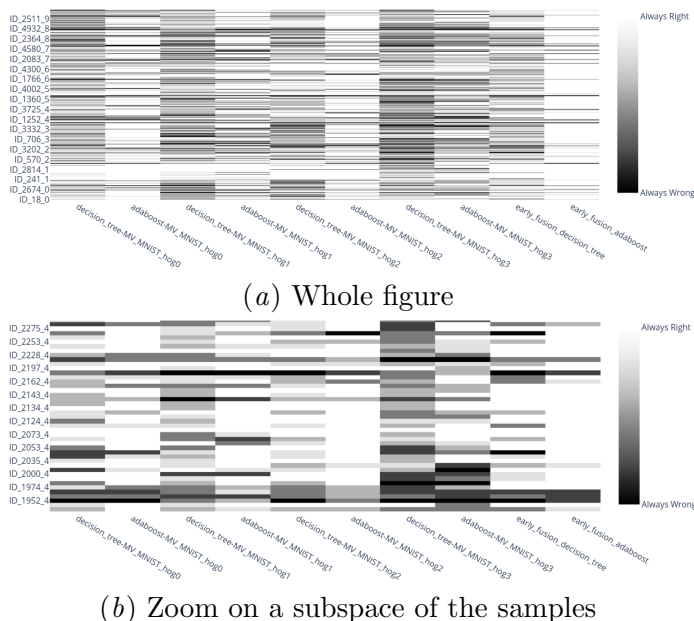


Figure 3: A static version of the interactive heatmap that shows the number of times that each classifier failed to classify a sample. A sample that has never been well-classified is shown in black, while one that always have been well-classified is plotted in white, with all the grey nuances between. Therefore, a dark row in the heatmap shows a sample that has been mainly mis-classified by the models.

view gathers 3 random orientations of a 12-orientations histogram of gradients. We call this new version MV_MNIST to avoid any confusion with the real MNIST.

3.1. A Simple Multi-view Experimental Protocol for Classification

Given a supervised classification task over a multi-view dataset, the first trivial usage of SuMMIT is to set up and run several learning processes, to be able to compare their classification performance and explain their success or failure.

To do so, many mono-view algorithms are built-in in SuMMIT, as it is coded in Python3 over the `scikit-learn` machine learning library. Therefore, SuMMIT integrates supervised algorithms and metrics from `scikit-learn` through a mapping table. Similarly, the early fusion versions of those off-the-shelf mono-view algorithms are included in SuMMIT, to allow a first level of multi-view analysis. In addition, a framework to build a late fusion classifier based on any `sklearn`-based models has been included in SuMMIT’s pool of multi-view classifiers. Finally, the algorithms provided by the `scikit-multimodallearn` library have been added to the multi-view collection, to provide purely multi-view models. Therefore, running a benchmark on a multi-view dataset may allow to build a baseline on the considered task, while providing a high-level understanding on the amount of information in each view, and the best way to decrypt it.



Figure 4: A static version of the interactive heatmap representing the sorted features by importance, for the algorithms that provide this insight.

Technically, the setting up is easily specified in a `yaml` configuration text file and the runs are executed locally, insuring data privacy. The results are locally saved in a dedicated folder, gathering an **interactive bar plot** for each required metric, represented in Figure 2 for the accuracy score. It shows the train and test scores that have been averaged across several train/test splits for each model, sorted by best mean test score. In addition, it provides a **heatmap** representing the **classification success** or failure of each classifier on each sample, shown in Figure 3. This kind of diagram allows to grasp a general point of view on the classifiers and views relevance on each sample. Indeed, in Figure 3(b), we see a zoomed-in version, that shows that Adaboost has trouble classifying the sample ID_3094_5 in every view, except `hog-1`. This type of graph allows to detect possible outliers in the samples, and provides an insight on the relevance of each view for each sample. It is completed by a **bar plot** showing the number of classifiers that failed to classify each sample, to ease the outlier detection. Moreover, the **feature importances** are described by a heatmap for each classifier, represented in Figure 4. This information derives from `sklearn`'s `feature_importances_` attribute that provide insight on how important each feature was in the classification process for each model. We generalized the concept to multi-view algorithms and plotted a heatmap that shows explainability for each model in the benchmark. Ultimately, a **text file** for each model learned, summarizes its scores, its confusion matrix, and interpretation for the models that provide such information.

As an illustration on MV_MNIST, let us suppose that the the user wants to compare the accuracy of a decision tree and Adaboost on each view, alongside their early fusion versions on all the views. This can be encoded in the `yaml` configuration file as:

```
name: ["multiview_mnist"] # Name of the dataset file
type: ["multiview", "monoview"] # Type of analysis
algos_monoview: ["decision_tree", "adaboost"] # Monoview algorithms
```



```
algos_multiview: ["early_fusion_adaboost", # Multiview algorithms
"early_fusion_decision_tree"]
```

The configuration file allows the user to indicate the performance measures to be considered, among the main scalar classification metrics.

Stepping into further in the integration functionalities of SuMMIT, several multi-view algorithms other than early and late fusion can in turn be specified as `algos_multiview`. That way, the relevance of these multi-view algorithms on the dataset can be compared to more traditional approaches. The following configuration file indicates that early and late fusion of Adaboost must be compared with the Mumbo algorithm (Koço and Capponi, 2011) according to `scikit-learn`'s accuracy and F1 scores.

```
type: ["multiview", "monoview"]
algos_monoview: ["adaboost"]
algos_multiview: ["early_fusion_adaboost", "late_fusion", "mumbo"]
metrics: # The metrics configuration
  accuracy_score: {} # Accuracy with default configuration
  f1_score: {} # F1-score with default configuration
```

3.2. Optimizing Hyper-parameters and Reproducible Results

As SuMMIT's main goal is to set baselines on new tasks, it includes two hyper-parameter optimization methods: a grid search and a randomized search (attribute `hps_type` of the configuration file), based on prior hyper-parameter distributions. Both methods are combined with k -fold cross-validation to validate the hyper-parameters. As this process can be costly for large datasets or time-consuming learning algorithms, SuMMIT outputs a report for each classifier, giving the best hyper-parameter's set, so one can re-use it without the optimization process in future benchmarks. These features are multi-view compatible generalizations of the existing code from `scikit-learn`. Moreover, to allow experts to provide insight for the most relevant hyper-parameter distributions, the hyper-parameters available for tuning are predefined for each integrated mono- and multi-view algorithm but can be overwritten in the configuration file.

SuMMIT is configured thanks to an easily shareable file which is saved in each result directory alongside a seed that control all the random number generators. As a consequence, SuMMIT allows to reproduce any benchmark by simply sharing the result directory and the dataset. For example, the following configuration calls a randomized search with 30 draws for each classifier with 5 folds cross-validation and a random seed of 42.

3.3. Plugging a New Multi-view Algorithm

Finally, it is simple to add one's own implementations to SuMMIT, allowing to challenge the baseline with one's own algorithm(s). Indeed, the provided classifier pool in the `master` branch is generic, but specific algorithms can be added to solve peculiar tasks (sparse decision functions, imbalanced datasets, etc.). To add a multi-view model to SuMMIT one must add a python file named after the classifier to `summit/multiview/multiview_classifier`, in which one has to implement a short adaptation class that inherits the mandatory `BaseMultiviewClassifier`, and that provides `fit` and `predict` methods as well as the hyper-parameter distributions for the classifier.

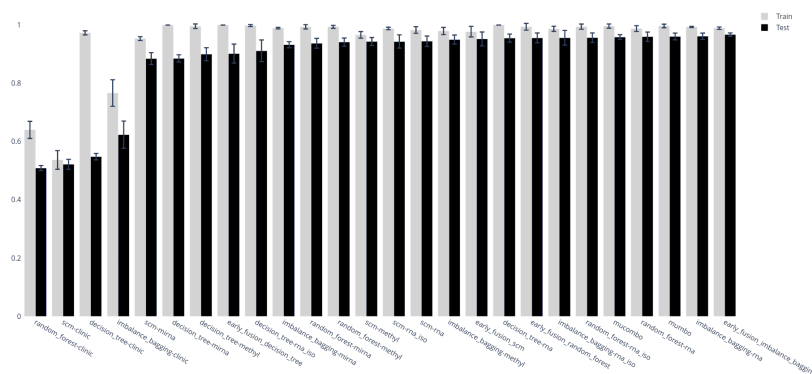


Figure 5: Balanced accuracies of the models learned. The vast majority of the models are relevant, except for the ones learned on the clinical view (the first four bars of this plot).

The modularity of SuMMIT is a huge advantage, allowing it to be included in anyone’s research, as long as the additional algorithms were conceived with the basic `sklearn` compatibility. An example of how to include a new multi-view classifier is given in Appendix ??, similarly for a hypothetical mono-view classifier in Appendix ?. We highly encourage the user to share their already published code, to durably include it in SuMMIT.

3.4. Real World Use Case : Multi-Omic Study

To demonstrate the usefulness of SuMMIT, and the information it outputs on a specific task, let us investigate the multi-omic dataset from a biological study (Osseni et al., 2021), comprised of 5 views. Each view describes the output of a sensor, or the answers on a clinical questionnaire, and we aim at predicting the type of breast cancer between Triple Negative (TNBC) and Non-TNBC for **902 patients**. The views available are (1) a **clinic** view that regroups **18 categorical features** about the patients, (2) an **RNA-iso** view, regrouping the expression levels of mRNA isoforms, (3) a DNA **methylation** view, describing the methylation levels of the DNA, (4) an **RNA** view, regrouping the Single Nucleotide Polymorphisms (SNPs), detecting mutations in the RNA, and (5) a **mi-RNA** view containing **250 features** describing the expression of micro RNA. Similarly to the study (Osseni et al., 2021), the larger views were cropped at **2000 features** using dimension reduction.

To analyze this dataset and establish baselines, we run SuMMIT, with four mono-view algorithms on each view, their early fusion versions and a purely multi-view algorithm on 5 train/test splits. To fit this particular imbalanced task, we added the `imbalance_bagging` from the `imbalanced-learn` library (Lemaître et al., 2017), and μ CoMBo, a multi-view version of CoMBo (Koço and Capponi, 2013), an algorithm that learns by minimizing the confusion matrix norm, specifically designed for multi-class and multi-view imbalanced tasks. The experiment took 1.5 hours, with more than 87% of this time used for hyper-parameter optimization on 6 parallel threads, leading to the fitting and testing of $((4 \times 5) + 5) \times 5 = 120$ models for which 6 random search draws were done (to reduce the computation time).

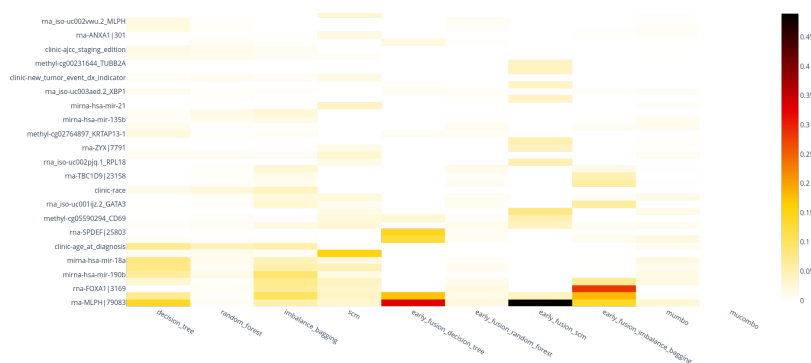


Figure 6: Feature importances of all the models learned on the multi-view dataset, sorted with the most important across all models at the bottom. This graph shows that there is a diversity in the features on which each classifier bases its model, but that a subset are used in several models with high importance.

The balanced accuracy results are given in Figure 5, in which we see that early fusion is sufficient to classify well the dataset, as the majority of the views are compatible data types, such as densities, or gene expression levels, all of them, transcribed by scalars. An interesting property to note is that the early fusion of the Random Forest has a lower score than the mono-view version on both the RNA ISO and RNA views. This strange behaviour might be caused either by overfitting, or by the fact that the Random Forest can be impacted by noisy features during its bagging process. In addition, we can see on Figure 5 that either the clinical view does not hold any information, or possibly we did not find the model to decrypt it, but all the mono-view algorithms failed on this task.

Similarly, the feature importance graph in Figure 6 shows that the FOXA1 gene is frequently used by the algorithms, and it is a known cancer-linked gene ⁷, still identified in very recent studies (Li et al., 2021), similarly for the MLPH gene that has been also linked to breast cancer (Thakkar et al., 2015). Therefore, with one run of SuMMIT, in 1.5 hours, we got relevant information about the learning task, the contribution of each model on each view. We found known biomarkers that validate our models. Our results can now be used by biologists for new tests or to establish new hypotheses.

3.5. Towards Additional Functionalities

SuMMIT’s development has been performed using continuous integration with Docker and automated tests, covering 93% of the code. It is available under MIT license on GitHub⁸ to allow collaborative development and hosting the automated Sphinx documentation and examples.

This paper presents the first release of SuMMIT. As a consequence, it comes with some limitations that could be alleviated through a deeper coupling with `scikit-learn` or other data science libraries. For example, similarly to `scikit-learn`, SuMMIT does not deal with missing values, which therefore must be imputed beforehand. More generally, SuMMIT does

7. <https://www.ncbi.nlm.nih.gov/gene/3169>

8. <https://github.com/multi-learn/summit>

not include feature pre-processing tools although such a functionality is clearly of interest in multi-view experiments, especially for improving the performances of early fusion with prior or posterior standardisation or component analysis. The architecture of SuMMIT is opened to such an integration. In addition, SuMMIT is currently dedicated to supervised classification. An upgrade for handling regression tasks is easy to produce as it would only require to add relevant performance measures and algorithms in the mapping tables.

The best way to improve SuMMIT is to build a user community that will add features based on their needs to collaboratively develop it in the most interesting direction. Indeed, SuMMIT is mainly used in a local environment and would profit from a larger user-base.

4. Conclusion and Future Work

SuMMIT is an easy handling platform that provides first insights in model selection for multi-view problems. Thanks to its plug-and-play architecture, it can be used to assess the relevance of innovative multi-view algorithms compared to usual approaches. Moreover, it is built on the robust `sklearn` framework and thanks to its public availability on GitHub and its Docker continuous integration, it can be collaboratively developed.

We developed a customizable multi-view generator compatible with SuMMIT to simulate specific problems, which integration in the platform is in progress, to provide generated datasets to the user. Moreover, we are currently working to improve SuMMIT on several levels, such as parallelization and GPU interface. To be more broadly used, it might also need a graphical interface, or pre-processing tools such as missing value imputation.

Acknowledgments

We thank Riikka Huusari for her stimulating help and the anonymous reviewers for their relevant and helpful comments. This work is supported by National Science and Engineering Research Council of Canada (NSERC) Discovery grant 262067, and granted by Lives Project (ANR-15-CE23-0026).

Importantly, we would like to acknowledge Pr. François Laviolette who passed away recently. He had a huge impact on our work and was a mentor. His guidance and feedback were important to us all.

References

- Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2018.
- Baptiste Bauvin, Dominique Benielli, Sokol Koço, Cécile Capponi, and François Laviolette. Multi-view artificial generation engine: Mage – controlled data generator for multi-view learning. June 2021.
- Dominique Benielli, Baptiste Bauvin, Cécile Capponi, Sokol Koço, Hachem Kadri, Riikka Huusari, and François Laviolette. Toolbox for Multimodal Learn (scikit-multimodallern). working paper or preprint, December 2021.
- Bokai Cao, Hucheng Zhou, Guoqiang Li, and Philip S. Yu. Multi-view machines. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*,

- WSDM '16, page 427–436, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450337168. doi: 10.1145/2835776.2835777.
- Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- Riikka Huusari, Hachem Kadri, and Cécile Capponi. Multi-view metric learning in vector-valued kernel spaces. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 415–424, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR.
- Sokol Koço and Cécile Capponi. A boosting approach to multiview classification with cooperation. volume 6912, pages 209–228, 09 2011. doi: 10.1007/978-3-642-23783-6_14.
- Sokol Koço and Cécile Capponi. On multi-class learning through the minimization of the confusion matrix norm. *Journal of Machine Learning Research*, 29, 03 2013.
- Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.
- Bin Li, Xiang Cheng, Ying Zhu, Hao Wan, Zequn Lu, Yimin Cai, Wenhui Li, Pengfei Yi, Li Liu, Jiang Chang, Xiaoping Miao, Jianbo Tian, and Rong Zhong. FOXA1 of regulatory variant associated with risk of breast cancer through allele-specific enhancer in the chinese population. *Breast Cancer*, 29(2):247–259, October 2021.
- Mazid Abiodoun Ossen, Prudencio Tossou, Jacques Corbeil, and François Laviolette. Applying pycmgrouper to breast cancer biomarkers discovery. In *BIOINFORMATICS*, pages 72–82, 2021.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- Ronan Perry, Gavin Mischler, Richard Guo, Theodore Lee, Alexander Chang, Arman Koul, Cameron Franz, Hugo Richard, Iain Carmichael, Pierre Ablin, Alexandre Gramfort, and Joshua T. Vogelstein. mvlearn: Multiview machine learning in python. *Journal of Machine Learning Research*, 22(109):1–7, 2021.
- Cees Snoek, Marcel Worring, and Arnold Smeulders. Early versus late fusion in semantic video analysis. pages 399–402, 01 2005. doi: 10.1145/1101149.1101236.
- Arvind Thakkar, Hemanth Raj, Ravishankar, Bhaskaran Muthuvelan, Arun Balakrishnan, and Muralidhara Padigar. High expression of Three-Gene signature improves prediction of Relapse-Free survival in estrogen Receptor-Positive and Node-Positive breast tumors. *Biomark Insights*, 10:103–112, November 2015.