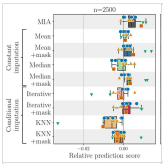


Jul 07, 2021

Benchmarking missing-values approaches for predictive models on health databases

Alexandre Perez-Lebel[1], Gaël Varoquaux[1], Marine Le Morvan[1], Julie Josse[2], Jean-Baptiste Poline[3]

[1]Inria, Palaiseau, France; [2]Inria, Montpellier, France; [3]McGill University, Montreal, Canada

Missing values analysis

Alexandre Perez-Lebel

ABSTRACT

BACKGROUND

As databases grow larger, it becomes harder to fully control their collection, and they frequently come with missing values: incomplete observations. These large databases are well suited to train machine-learning models, for instance for forecasting or to extract biomarkers in biomedical settings. Such predictive approaches can use discriminative --rather than generative-- modeling, and thus open the door to new missing-values strategies. Yet existing empirical evaluations of strategies to handle missing values have focused on inferential statistics.

RESULTS

Here we conduct a systematic benchmark of missing-values strategies in predictive models with a focus on health databases: two electronic health record datasets, a population brain imaging one, and a health survey. Using gradient-boosted trees, we compare native support for missing values with simple and state-of-the-art imputation prior to learning. We investigate prediction accuracy and computational time. For prediction after imputation, we find that adding an indicator to express which values have been imputed is important, suggesting that the data are missing not at random. Elaborate missing values imputation can improve prediction compared to simple strategies but requires longer computational time on large data. Learning trees that model missing values --with missing incorporated attribute-- leads to robust, fast, and well-performing predictive modeling.

CONCLUSIONS

Native support for missing values in supervised machine learning predicts better than state-of-the-art imputation with much less computational cost. When using imputation, it is important to add indicator columns expressing which values have been imputed.

PROTOCOL CITATION

Alexandre Perez-Lebel, Gaël Varoquaux, Marine Le Morvan, Julie Josse, Jean-Baptiste Poline 2021. Benchmarking missing-values approaches for predictive models on health databases. **protocols.io** https://protocols.io/view/benchmarking-missing-values-approaches-for-predict-bmvgk63w

KEYWORDS

Missing Values, Machine Learning, Supervised Learning, Benchmark, Imputation

CREATED

Sep 29, 2020

LAST MODIFIED

Jul 07, 2021

PROTOCOL INTEGER ID

42632

GUIDELINES

This protocol details the experiments run in the article *Benchmarking missing-values approaches for predictive models on health databases*, Perez-Lebel et al. 2021. Documented code is available on GitHub.

Accessing the databases can be time consuming. We published our detailed results in a CSV file to allow further analysis without needing to access the data.

MATERIALS TEXT

Computing cluster

SAFETY WARNINGS

No safety warnings.

Introduction

1 This protocol details the experiments run in the article *Benchmarking missing-values approaches for predictive models on health databases*, Perez-Lebel et al. 2021. Documented code is available on GitHub:

Benchmarking missing-values approaches for predictive models...
source by Alexandre Perez-Lebel

And can be installed through the following steps:

Install

```
git clone https://github.com/alexprz/article-benchmark_mv_approaches.git
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

Download and install the code reproducing the experiments.

We benchmarked two sets of 9 supervised predictive methods on 13 prediction tasks taken from 4 health databases.

Data

2 Each one of the 4 databases needs to be downloaded separately from their respective source project. Access to Traumabase, UK BioBank and MIMIC-III, requires an application. NHIS is freely available. Once downloaded, data path of each database can be updated in the TB.py, UKBB.py, MIMIC.py and NHIS.py files which are in the *database/* folder of the project.

2.1

Traumabase

The Traumabase Group (TB) is a collaboration studying major trauma. The database gathers information from 20 French trauma centers on more than 20 000 trauma cases from admission until discharge from critical care. Data collection started in 2010 and is still ongoing in 2020. We used records spanning from 2010 to 2019. We defined 5 prediction tasks on this database, 4 classifications and 1 regression.

Data can be obtained by contacting the team on the Traumabase website.

2.2

UKBB

UK Biobank (UKBB) is a major prospective epidemiology cohort with biomedical measurements. It provides health information on more than 500 000 United-Kingdom participants aged between 40 to 69 years from 2006 to 2010. We defined 5 tasks on this database, 4 classifications and 1 regression.

The data are available upon application as detailed on the UK BioBank website.

2.3

MIMIC-III (v1.4)

The Medical Information Mart for Intensive Care (MIMIC) database is an Intensive Care Unit (ICU) dataset developed by the MIT Lab for Computational Physiology. It comprises deidentified health data associated with about 60 000 ICU admissions recorded at the Beth Israel Deaconess Medical Center of Boston, United States, between 2001 and 2012. It includes demographics, vital signs, laboratory tests, medications, and more. We defined 2 classification tasks on this database.

The data can be accessed via an application described on the MIMIC website. Note that, as of the time of writing, the completion of an online MIT course is required for the application. We used the 1.4 version of the data in the project.

2.4

NHIS (2017)

The National Health Interview Survey (NHIS) is a major data collection program of the National Center for Health Statistics (NCHS), part of the Centers for Disease Control and Prevention (CDC) in the United States. It aims to monitor the health of the population. Since 1957, it collects data from United-States population. We used the 2017 edition, summing up to approximately 35 000 households containing about 87 500 persons. We defined 1 regression task on this database.

It is freely-accessible on the NHIS website.

Prediction tasks

3 From these databases, we defined 13 prediction tasks. That is, a set of input features and an outcome to predict. All features of each task belong to the same database.

Available tasks can be obtained with:

Available tasks

python main.py info available -t

List the names of all the available tasks.

Names of the available tasks are:

TB/death_pvals

TB/platelet_pvals
TB/hemo
TB/hemo_pvals
TB/septic_pvals
UKBB/breast_25
UKBB/breast_pvals
UKBB/skin_pvals
UKBB/parkinson_pvals
UKBB/fluid_pvals
MIMIC/septic_pvals
MIMIC/hemo_pvals
NHIS/bmi_pvals

Predictive methods

4 36 predictive methods are available. The list of their IDs and names can be obtained running:

Available models

python main.py info available -m

List the IDs and names of all the available methods.

IDs and names of the available methods are:

0: Classification
1: Classification_Logit
2: Regression
3: Regression_Ridge
4: Classification_imputed_Mean
5: Classification_Logit_imputed_Mean
6: Regression_imputed_Mean
7: Regression_Ridge_imputed_Mean
8: Classification_imputed_Mean+mask
9: Classification_Logit_imputed_Mean+mask
10: Regression_imputed_Mean+mask
11: Regression_Ridge_imputed_Mean+mask
12: Classification_imputed_Med
13: Classification_Logit_imputed_Med
14: Regression_imputed_Med
15: Regression_Ridge_imputed_Med
16: Classification_imputed_Med+mask
17: Classification_Logit_imputed_Med+mask
18: Regression_imputed_Med+mask
19: Regression_Ridge_imputed_Med+mask
20: Classification_imputed_Iterative
21: Classification_Logit_imputed_Iterative
22: Regression_imputed_Iterative
23: Regression_Ridge_imputed_Iterative
24: Classification_imputed_Iterative+mask
25: Classification_Logit_imputed_Iterative+mask
26: Regression_imputed_Iterative+mask
27: Regression_Ridge_imputed_Iterative+mask
28: Classification_imputed_KNN
29: Classification_Logit_imputed_KNN
30: Regression_imputed_KNN
31: Regression_Ridge_imputed_KNN
32: Classification_imputed_KNN+mask

33: Classification_Logit_imputed_KNN+mask
34: Regression_imputed_KNN+mask
35: Regression_Ridge_imputed_KNN+mask

Feature selection

5

11 tasks have their features automatically selected with a simple ANOVA-based univariate test of the link of each feature to the outcome (task name ends with "_pvals" in the code and "_screening" in the article).
The 2 remaining tasks have their feature manually defined following the choices of experts in prior studies.

5.1 ANOVA-based feature selection

Categorical features are first one-hot encoded. Then, the ANOVA-based univariate test is performed on one third of the samples which are then discarded. We kept the 100 encoded features having the smallest 100 p-values. Once the features are selected, the cross-validated prediction is performed on the remaining two thirds of the samples.
For these tasks, there are 5 trials during which the samples on which the selection test is performed are redrawn, and the prediction each time fitted on the new remaining samples and the new selected features.

We used *f_classif* and *f_regression* from the *feature_selection* module of scikit-learn.

For each of these tasks, p-values of the test can be computed for each each trial by running:

Feature selection

python main.py select {task_name} --T {T}
Compute p-values of ANOVA-based test to select features

Be careful to replace placeholders {task_name} and {T} by the name of the task and the trial ID (0 to 4) respectively.
Example:

Example of feature selection

python main.py select TB/death_pvals --T 0
Compute p-values of ANOVA-based test to select features of the task TB/death_pvals on the first trial.

5.2 Manual selection following experts

Features for the hemorrhagic shock prediction (task named TB/hemo) in the Traumabase database are defined following Jiang et al.:

Wei Jiang, Julie Josse, Marc Lavielle, TraumaBase Group (2020). Logistic Regression with Missing Covariates – Parameter Estimation, Model Selection and Prediction within a Joint-Modeling Framework. Computational Statistics and Data Analysis. http://10.1016/j.csda.2019.106907

Features for the breast cancer prediction (task named UKBB/breast_25) are defined following Läll et al.:

> Kristi Läll, Maarja Lepamets, Marili Palover, Tõnu Esko, Andres Metspalu, Neeme Tõnisson, Peeter Padrik, Reedik Mägi, Krista Fischer (2019). Polygenic prediction of breast cancer: comparison of genetic predictors and implications for risk stratification. BMC Cancer.
> http://10.1186/s12885-019-5783-1

There is only 1 trial for these tasks.

Prediction 71w 3d

6 71w 3d

Scale
To study the influence of the scale on the results, we decided to work on 4 sizes of the training set: 2 500, 10 000, 25 000 and 100 000. For each one of these sizes are run the following operations.

Outer cross-validation
First, 5 train sets are randomly selected with the appropriate number of samples (2 500, 10 000, 25 000 or 100 000). For each train set, the test set is composed of all the remaining examples. Note that the size of the test set is considerably larger with a train set of 2 500 samples than with 100 000.

For each of the 5 folds, the methods are fitted on the train set and tested on the test set. For imputation-based methods, missing values of both train and test sets are first imputed with the imputer fitted on the training set only. Then, a cross-validated hyper-parameter tuning of the predictive model's parameters is performed on the imputed train set. Once the best model is obtained, it is tested against the imputed test set and its score is reported (accuracy or R^2). Methods that do not require imputation follow the same pipeline with the imputation step skipped.

To draw the 5 folds, we used *StratifiedShuffleSplit* (resp. *ShuffleSplit*) from scikit-learn for classifications (resp. regressions).

Evaluating a method on a prediction task is done by running:

Prediction

python main.py predict {task_name} {method_id} --T {T}
Benchmark a method on a prediction task

Be careful to replace placeholders {task_name}, {method_id} and {T} by the name of the task, the ID or name of the method and the trial ID (0 to 4) respectively.
Example:

Prediction example

python main.py predict TB/death_pvals 0 --T 0
Benchmark method with ID 0 on the task TB/death_pvals on the trial 0.

Results are dumped in the *results/* folder.

12000:00:00 CPU hours to run the full benchmark

6.1 Imputation

4 imputation methods are available:

- Imputation with the mean.
- Imputation with the median.
- Iterative imputation.
- Imputation with the nearest neighbors.

For each of them, new binary features can be added to the data. This binary mask encodes whether a value was originally missing or not.

The imputer is fitted on the train set only and both the train and test sets are then imputed with the fitted imputer. Doing so avoids leaking information from the train set to the test set and then helps to avoid overfitting.

We used *SimpleImputer*, *IterativeImputer* and *KNNImputer* from scikit-learn.

6.2 Cross-validated hyper-parameters tuning

Hyper-parameters of the predictive models are tuned in a cross-validated manner on the train set (imputed or not).

We used *GridSearchCV* from scikit-learn to perform the cross-validated hyper-parameters tuning. The cross-validation folds are drawn with *StratifiedShuffleSplit* (resp. *ShuffleSplit*) from scikit-learn for classifications (resp. regressions).

Results

7

Once the results of all the methods are obtained, they are gathered in a single CSV file using the following command:

Aggregate results

python main.py aggregate results/

Merge all results in a single csv file

This creates a *scores.csv* file in the *scores/* folder.

The aggregated results obtained during our experiment are given in our repository. This allows to reproduce the figures and tables and to analyze further the results without needing the original data.

Figures and tables

8

Most of the figures and tables of our article can be easily reproduced without requiring the original data (based on saved results only).

Reproduce figures and tables from aggregated scores

python main.py figs -h

Helper to the command that reproduces the figures and tables of our article based on the aggregated scores.

Some figures however need the data to be build.

Reproduce figures about databases

python main.py datastats -h

Helper to the command that reproduces the figures about databases' statistics

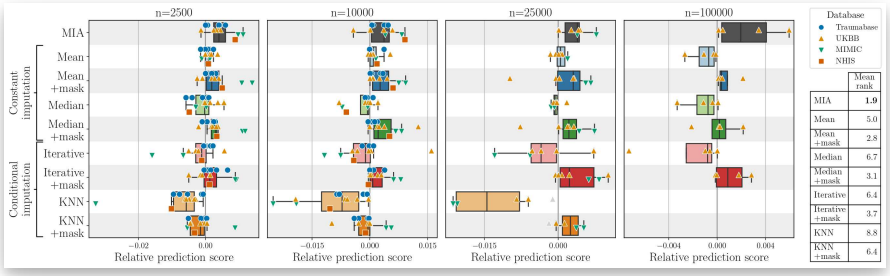
All figures and tables are saved in the *graphics/* folder of the repository.

The main figure can be reproduced with:

Main figure

python main.py figs scores

Reproduce the main figure of the article.



Main figure: Comparison of prediction performance and training times across the 9 methods for 13 prediction tasks spread over 4 databases, and for 4 sizes of dataset (2 500, 10 000, 25 000 and 100 000 samples).