IFC-Bank of Italy Workshop on "Data Science in Central Banking: Applications and tools"

14-17 February 2022

gingado: a machine learning library focused on economics and finance[1]

Douglas Araujo,
BIS

[1] This contribution was prepared for the workshop. The views expressed are those of the authors and do not necessarily reflect the views of the Bank of Italy, the BIS, the IFC or the other central banks and institutions represented at the event.

gingado: a machine learning library focused on economics and finance

Douglas K. G. Araujo

Abstract

gingado is an open source Python library that offers a variety of convenience functions and objects to support usage of machine learning in economics research. It is designed to be compatible with widely used machine learning libraries. gingado facilitates augmenting user datasets with relevant data directly obtained from official sources by leveraging the SDMX data and metadata sharing protocol. The library also offers a benchmarking object that creates a random forest with a reasonably good performance out-of-the-box and, if provided with candidate models, retains the one with the best performance. gingado also includes methods to help with machine learning model documentation, including ethical considerations. Further, gingado provides a flexible simulatation of panel datasets with a variety of non-linear causal treatment effects, to support causal model prototyping and benchmarking. The library is under active development and new functionalities are periodically added or improved.

Keywords: machine learning, open source, data access, documentation

JEL classification: C87, C14, C82

Contents

gingado: a machine learning library focused on economics and finance 1

1. Introduction 3
2. Data augmentation 5
   - 2.1 Basic data augmentation process 6
   - 2.2 Data augmentation with SDMX 7
   - 2.3 Is it worth adding more and more data? 8
3. Automatic benchmark 10
   - 3.1 Other use cases for a benchmark model 11
   - 3.2 Custom automatic benchmarks 12
4. Real and synthetic datasets 12
   - 4.1 Real datasets 12
   - 4.2 Simulated datasets 13
5. Model documentation 15

5.1 Ethical issues in economics and finance for ML models 18

6. What to expect next? 20

7. Conclusion 20

References 23

1. Introduction

Conceptually, every machine learning (ML) application entails the combination of a specific dataset, algorithm, cost function and optimisation procedure - and each of these components can be replaced more or less independently from the others (Goodfellow, Bengio, and Courville (2016)). The set of possibilities is particularly wide in many economics and finance use cases (Athey and Imbens (2019), Mullainathan and Spiess (2017), Varian (2014), Doerr, Gambacorta, and Garralda (2021) Chakraborty and Joseph (2017)). And there is often no definite answer as to which alternative is best (Mullainathan and Spiess (2017)). Hence, creating a ML application in practice can require multiple iterations and attempted improvements to achieve a satisfactory result. In fact, as of writing this paper, different steps of the process of creating ML models in economics and finance could be more streamlined, ranging from selection and use of the dataset or simulation of a dataset with known generating process, comparison of different ML models, and crucially, also the model's documentation.

gingado is a free, open source library that seeks to facilitate these and other steps in the use of ML in economics and finance in academic and practitioner settings, while promoting good modelling and documentation practices.[1] It offers a number of main contributions. First, gingado facilitates data augmentation of the original user dataset with statistical series from official sources, in a way that is relevant for each case and empirically testable on its ability to improve the model's performance. Second, gingado provides automatic benchmark models that perform well on a broad set of situations and that can train quickly to achieve a reasonable if not the best performance for the dataset at hand; users can also make use of a generic benchmark object to create their own automatic benchmarks. Third, for when the user needs to test a causal inference models, or benchmark existing models, gingado offers functions that flexibly simulate panel data with customised data generating processes that include linear and non-linear interactions and diverse treatment assignment mechanisms and (homogenous or heterogenous) treatment effects. Fourth, gingado promotes documentation of the ML model as part of the development workflow, automating some documentation steps to leave users room for concentrating on more value-added documentation items such as ethical considerations. Also here gingado offers users the possibility for users to create their own model documentation templates that are easy to embed in the modelling practice. And fifth, gingado offers a number of other utilities for helping data science work in economics, in particular in time series of panel data.

[1] gingado's instructions, documentation, practical examples and source code are available at https://dkgaraujo.github.io/gingado/. The library is named after the Brazilian concept that is difficult to translate but broadly represents a dance-like non-stop swing of the body that is often associated with flexibility to adjust oneself to adverse situations, eg in life or in a football match. The name is also as an homage to the Afro-Brazilian martial art of Capoeira, where having "gingado" is key. I chose "gingado" as the name for this library focused on using ML in economics and finance because it combines the idea of a constant swing, akin to how economic and financial series also "swing" around a trend through the course of business and financial cycles, with flexibility, similar to how ML models are considerably flexible when fitting functional forms.

gingado is in active development. The library follows three design principles:

1. **Flexibility**: gingado works well out of the box but users can customise its objects in ways that are more suitable for their use cases;

2. **Compatibility**: gingado works seamlessly with other widely used libraries in data science and ML; and

3. **Responsibility**: gingado promotes model documentation and ethical considerations as key steps in the modelling process.

In addition, gingado seeks to be a parsimonious library that complements, rather than redoes, existing functionalities of other widely used libraries. gingado's application programming interface (API) is compatible in particular with scikit-learn (Pedregosa et al. (2011)), which itself is the basis for a variety of other ML libraries, and can be adapted to work with other Python ML libraries with minimal adjustment. In addition, the gingado API can be generally accessed from R (via the integration with Python with the reticulate package, Ushey, Allaire, and Tang (2022)) or from other languages or environments such as Stata and MatLab. The library can be used in a modular way: users might prefer to use gingado only to augment their datasets, create automatic benchmark models, simulate causal datasets or document their models, or any combination thereof.

These characteristics make gingado a potentially useful tool across many domains in economics and finance. ML algorithms are already amply used in economics for prediction problems (in the sense of Kleinberg et al. (2015)), causal inference[2] (eg, Chernozhukov, Demirer, et al. (2018) and Athey, Tibshirani, and Wager (2019)), model estimation (Maliar, Maliar, and Winant (2021), Fernández-Villaverde, Hurtado, and Nuno (2019) and Duarte (2018)), estimation of models with non-traditional data (Ferreira et al. (2021)) and even being the subject of study themselves (Fuster et al. (2022), Giannone, Lenza, and Primiceri (2021)). gingado's functionalities can be used as appropriate in each instance. Central banks have also been using ML in a variety of applications (Araujo et al. (2023)), and the practitioner use cases in the industry are numerous and diverse.

To showcase practical applications, the online documentation includes two end-to-end examples[3] (with more to come over time): the automatic benchmark and model documentation functionalities are illustrated with the cross-country dataset with over 60 variables used to analyse drivers of GDP growth per capita (Barro and Lee (1994)). The dataset was also recently used by Giannone, Lenza, and Primiceri (2021) to study the different predictive abilities of sparse vs dense models. Another example focuses on attempts to forecast exchange rates (Rossi (2013)), illustrating how gingado's utilities can be used to compare different lags of the model, download specific data from SDMX sources, augment the original dataset with other relevant data, quickly create a benchmark model and use it compare different alternatives, and finally how to document the model to promote responsible model maintenance and usage. The examples illustrate ways in which gingado can help economists' workflow.

The next section describes how gingado facilitates data augmentation. Section 3 outlines the automatic benchmark process, followed by a discussion of real and

[2] A recent compilation of causal inference techniques from various domains is https://neurips.cc/virtual/2021/workshop/21871.

[3] Available at: https://dkgaraujo.github.io/gingado/.

simulated data functions in Section 4. Section 5 describes gingado's model documentation framework. The last section concludes. The online documentation describes how to install the most up-to-date version of gingado. More advanced topics like how to customise automatic benchmark models and model documentation templates can also be found online.

2. Data augmentation

Publicly available data have a long tradition in economics and finance research and practice. For example, the Federal Reserve Bank of St. Louis' Federal Reserve Economic Data (FRED) system has developed in tandem with wider adoption of the internet itself (see Stierholz (2014) for an interesting narrative of FRED's history). Similarly, numerous other central banks, statistical agencies and international financial institutions put datasets in the public domain in one form or the other. A number of statistics organisations created in the early 2000's an initiative to promote the collection, compilation and dissemination of statistical data, the Statistical Data and Metadata Exchange (SDMX), which is now in its version 3.0.[4] In addition, other data aggregators such as Base dos Dados and DBnomics host an incredible amount of economic and financial series.

Many of these services allow users to access data in a programmatic way, ie setting up a computer programme to download the data instead of the user manually accessing the website, selecting the data, downloading it in a file and incorporating the data in the analyses. Thanks to SDMX and to the broader availability of user-friendly data APIs like the one offered by FRED, querying data from trusted sources to augment the user's original dataset has become much easier than in the past. This allows more research to benefit from the main advantages of programmatic data access as opposed to manual downloads: the data acquisition is reproducible, auditable and scalable. Accessing data programmatically also allows any numeric transformations, consistency checks and data imputation routines that are applied on the dataset to be done in a reproducible and transparent way.

In addition, programmatic access to data can also ensure that any data that are added to the original dataset are done so in a consistent way. For example, SDMX includes the concept of "codelists", which are standardised definitions that apply across dataset domains. One specific codelist contains all possible realisations of the frequency of a dataset (ie, daily, weekly, monthly, etc.), and another codelist encompasses all possible codes for specific currencies. This technology ensures that the user has greater control over the datasets to be incorporated.

The considerations above are even more important when models are in the production stage. In economics and finance, ML models are occasionally designed to be run in production settings instead of a one-off execution; and this is often performed by users that were not involved during development and therefore are not as familiar with the model's internal functioning. For example, a model forecasting stock prices might be used extensively over time - by stock traders or portfolio managers, not the developers. These situations tend to take place in situations where

[4] A technical description of version 3.0 is found here: https://sdmx.org/wp-content/uploads/SDMX_3-0-0_SECTION_1_FINAL-1_0.pdf.

the ML model will require future observations of the datasets used for training. Therefore, automating the process of data augmentation in a way that can ensure consistency between datasets used for training and later for inference helps to insulate users from worrying about these processes related to the use of additional data.

gingado aims to facilitate this process of finding and adding new datasets, leveraging the public availability of reliable data from official sources and automatically ensuring that the augmented dataset is consistent with the original, by having the same frequency and time period. With this, the same publicly available dataset(s) used during model training are downloaded and used each time with the appropriate time period when the model is run in the future, potentially by other people or automatically by systems. The current version of gingado achieves this by means of its data augmentation object `AugmentSDMX`; the idea is to gradually add to the public codebase other "data augmenters" that can scan and download publicly available datasets in a way that is consistent with the original dataset.

2.1 Basic data augmentation process

Similar to other parts of gingado, the API for data augmentation is designed for compatibility with scikit-learn. Specifically, the user decides which specific sources and dataflows will be scanned for data upon instantiation of the data augmenter object. Until then, the object does not perform any activity other than to store this and other parameters. It is only when one of the methods fit, transform, or fit_transform are called that the data augmenter will (a) read characteristics of the user's data and (b) search in the named sources and dataflows for datasets that correspond to the same frequency and time period. This process is shown in Listing 1, which assumes the user already has a data frame called X_train with training data *indexed by time*: gingado uses this time period information behind the curtains to obtain data of the relevant frequency and time periods. In the listing, the user first imports the class AugmentSDMX, and then define a dictionary `src` to list the SDMX sources and their relevant dataflows - more on how to find the sources below. The data augmenter in this example is set to look for the Composite Indicator of Systemic Stress (CISS) by the European Central Bank (ECB) and the central bank policy rates dataflow from the Bank for International Settlement (BIS). In the next line, an instance of the class AugmentSDMX is created using those sources listed in src, and the method fit_transform learns the frequency and time periods from X_train and use that knowledge to fetch all data series from these two sources that comply with these time requirements.

Basic example of the AugmentSDMX application programming interface

Python environment — Listing 1

```
from gingado.augmentation import AugmentSDMX
src = {'ECB': ['CISS'], 'BIS': ['WS_CBPOL_D']}
augmented_X_train = AugmentSDMX(sources=src).fit_transform(X_train)
```

Sources: Author.

The process described above will result in augmented_X_train, a dataset that contains the original data, now augmented by potentially numerous other series. The data are indexed by time, and for this reason every combination of permissible codelists in the augmented dataset will create a different variable (ie, a different column). For example, even if the original dataset represents only data from Brazil, the augmentation process shown in Listing 1 I will append the CISS for every country in the list as a different column, as well as the central bank policy rates. gingado makes an explicit choice to allow for this, instead of filtering by individual (in this example, retrieving only the dataset about Brazil), because the newly added data can have some level of information on the target variable, and if that is the case the model would probably uncover it. Of course, users that desire to add only data relating to a country, currency, etc might add those as relevant filters, as described in more detail in the documentation.

One important note is that the user remains responsible for ensuring that the data being automatically added is not a covariate that will interfere with the desired statistical properties of the task at hand. In other words, if the economist running the model is interested in anything more than just a predictive exercise without any statistical properties, then greater attention to the potential covariates being added is warranted. For example, ensuring that the covariate is not endogenous or otherwise a bad control (Hünermund, Louw, and Caspi (2023) discuss the sensitiveness of some ML-based inference to inclusion of bad controls.) In some settings, one strategy to ensure only adequate covariates are included is to consider well which data sets will be searched for by gingado: ie make sure to include only those broad groups of data that will not negatively interfere with the model inference.

2.2 Data augmentation with SDMX

gingado explores and fetches available datasets from official sources to augment user data using SDMX, an initiative by international organisations (BIS, ECB, Organisation for Economic Cooperation and Development - OECD, International Monetary Fund - IMF, World Bank - WB, Eurostat, and United Nations - UN) that develop and publish statistics from these sources. Since its inception in 2001, SDMX has grown to be used by other sources as well - primarily central banks and statistics agencies - as a standard to disseminate their data.

Downloading data from SDMX sources can be advantageous to users because of the variety of sources and the consistency of the concepts describing the datasets. Instead of dealing with the specific data descriptions and download processes of each of the SDMX participant institutions, users can rely on an API based on standardised concepts to fetch the data. For example, using SDMX to look across multiple sources for data at quarterly frequency related to the countries of Argentina, Brazil and South Africa for the period spanning the first quarter of 2015 to the fourth quarter of 2021 would use the codelists (introduced above) for frequency and for reference area. These are the same across the SDMX sources and dataflows, which facilitates the process of finding relevant datasets.

AugmentSDMX is currently based on the pandasdmx python package. The backend code searches all listed SDMX sources in this package, and retrieves the dataflows for those it is able to get (some sources may time out). Given that downloading all the relevant data from all sources could be a time-consuming operation, users define the SDMX source(s) from which to get the data. For each

source, the user might define the specific dataflows or ask for all dataflows of that source.

As of the time of writing, the data providers available for automatic download of data using AugmentSDMX are:[5]

- Australian Bureau of Statistics
- BIS
- Countdown 2030
- Deutsche Bundesbank (Germany)
- ECB
- Eurostat
- International Labour Organization - ILO
- Internationa Monetary Fund - IMF
- National Institute of Statistics and Geography (Mexico)
- National Institute of Statistics and Economic Studies (France)
- National Institute of Statistics (Italy)
- National Institute of Statistics (Lithuania)
- Norges Bank (Norway)
- National Bank of Belgium
- Organisation for Economic Cooperation and Development -OECD
- Pacific Data Hub
- Statistics Estonia
- United Nations Statistics Division
- UN International Children's Emergency Fund (UNICEF)
- World Bank Group's "World Integrated Trade Solution"
- World Bank Group's "World Development Indicators"

Each of those sources offers a number of dataflows, which are closely related datasets. For example, one dataflow related to foreign exchange rates could include the time series of multiple individual exchange rate pairs, and each pair can be downloaded in their nominal or real exchange rate. The dataflows from all of these sources could be available to train the ML model at hand. In total, the sources listed above result in 9,110 dataflows.

2.3 Is it worth adding more and more data?

While an increasing amount of data can lead to better results by ML models, there are situations where users might want to consider limiting the amount of data being fed to a model. The computation time might increase as the number of SDMX series are added, which is an important consideration as models in production also need to

[5] Users can get the up-to-date list with the gingado method `list_SDMX_sources` in `gingado.utils`.

look at the same variables used during training. Depending on how time-consuming downloading this new data is, having bigger datasets could significantly affect computation speed at run time. However, for cases where immediate response is not required, a bigger quantity of data could be considered reasonable, especially if it leads to a marked performance improvement. Therefore, the answer to this subsection's title will depend on each use case, and gingado can help the user answer it.

AugmentSDMX's API is compatible with scikit-learn in a specific way that allows it to be included in a pipeline. Pipelines are objects that apply a specific sequence of transformations on data, and possibly a final step consisting of an estimator (such as a predictor). These pipelines exist to compare sequences of steps as a whole with proper cross-validation, and to allow for a consistent way to estimate different versions of the same model without losing control of the data pipeline.

What this means in practice is that the user can apply standard parameter search algorithms such as grid search to test whether or not the inclusion of a particular dataset will impact the model, eg by improving its performance, changing the importance of regressors, etc. This process is exemplified in Listing 1: the user created with a few lines a model that, when fitted, will estimate two versions of the model: one without augmentation (ie, will "pass through" AugmentSDMX) and another with augmentation. The first four lines import the necessary objects from gingado and scikit-learn. Then, an instance of the data augmenter is created in the variable sdmx, in this case using all dataflows made available by the BIS. Note that unlike Listing 1, neither the method fit nor fit_transform were called yet and therefore no calculation or data download is done at this stage. In the next step, an instance of a Pipeline object is created - but still not fitted. This pipeline chains together two steps: the data augmentation and a random forest. Finally, a dictionary with a grid of parameters to be tested empirically and grid, a variable that performs an ML calibration using grid search with cross-validation, are created. param_grid is the key part of this listing, since it tells grid to test the two versions of the ML model: one that bypasses the augmentation step and uses only the user-provided dataset, and the other one that uses the sdmx variable to augment automatically the user dataset with all relevant and compatible (ie, with the same frequency and time periods) data from the BIS. When fitted using training data on covariates and dependent variables, grid will select the parameters in param_grid that result in models with the best performance. Thus the question of whether or not to use more data can be answered empirically and in a completely data-driven way. Beyond that, the user can even jointly search richer combinations of different parameters governing data augmentation, data transformation and model estimation by combining AugmentSDMX with scikit-learn's Pipeline.

Use of AugmentSDMX in a scikit-learn pipeline

Python environment — Listing 2

```
from gingado.augmentation import AugmentSDMX
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
```

```
sdmx = AugmentSDMX(sources={'BIS': 'all'})

pipeline = Pipeline([
    ('augmentation', sdmx),
    ('forest', RandomForestRegressor())
])
param_grid = {'augmentation': ['passthrough', sdmx]}
grid = GridSearchCV(
    estimator=pipeline,
    param_grid=param_grid
    )
```

Sources: Author.

3. Automatic benchmark

gingado offers users the possibility of automatically developing a benchmark ML model fine-tuned for their particular case, in a short time and with no input from the user other than the data (although users can also fine tune all aspects of the benchmark). This is achieved by means of an embedded parameter grid search mechanism that evaluates different versions of the underlying algorithm on the user's data, and selects that one that performs better as the benchmark.

The objective is to help the user during the exploratory phase of the development of a ML model. The practice of establishing a baseline model is common in ML practice. Without a goalpost, a baseline model that is relatively simple to understand and benchmark against, it can be difficult in practice to realise if one's model is actually performing well or just improving from a low base. So gingado allows users to quickly create a fully functioning model that can serve as benchmark, as shown in Listing 3. Similar to Listing 1, this listing assumes the user already has available two data frames: X_train, containing a panel of covariates, and y_train, which is the dependent variable. The first line imports the object ClassificationBenchmark. In the second line, an instance called benchmark is created and already fitted in the same line. Behind the scenes, this instance of ClassificationBenchmark will perform a grid search using a random forest with default parameters that tend to work well in my experience in a variety of datasets that tend to be in line with the size of empirical papers in economics. The fitted object then represents the model with the best performance, and can be used by the user for prediction, etc. Naturally, the user can pass other estimators and also other sets of parameters for the grid search, illustrating gingado's combination of convenience with flexibility.

Creation of an automatic benchmark model

Python environment

Listing 3

```
from gingado.benchmark import ClassificationBenchmark

benchmark = ClassificationBenchmark().fit(X_train, y_train)
```

Sources: Author.

The off-the-shelf implementations of this automatic benchmark model are based on random forests (Breiman (1996), Breiman (2001)), with one type for regression tasks and another one for classification. Random forests, one of the most widely used ML methods according to industry practitioners (Howard and Gugger (2020a)) present several advantages that make them good candidates for benchmark models. They tend to have a very good out-of-sample fit across a wide variety of data generating processes, and require little data preparation work compared to other ML models. Random forests also provides an intuitive way to measure the individual importance of regressors, although they don't lend themselves to interpreting the channels by which the regressors contribute to the prediction (Varian (2014)). These importance measures are the mean reduction in impurity occurring in the trees that use that particular variable. The values are then typically scaled so that the sum of all feature importance measures is always one. Practitioners use this measurement as one possibility for variable selection (Géron (2019), Kohlscheen (2021)).

Whenever a benchmark model is fitted to the data, the model automatically jumpstarts its documentation from a template and automatically filling some information on the model. In the off-the-shelf implementation, this documentation is done via the ModelCard object, described in more detail in section 5. From then on, the model documentation can be accessed - and most importantly, filled out by the user.

Benchmark models also have a specific method to compare themselves with other candidate models. This allows users to directly compare their own candidate models with the existing benchmark. When this is done, via the module `compare`, gingado also includes amongst the candidates to be tested an ensemble combination of all the candidates and the current benchmark. The inclusion of this candidate ensemble serves to leverage the advantage that ensemble models have over simpler models (Giannone, Lenza, and Primiceri (2021)).

3.1 Other use cases for a benchmark model

In addition to serving as a baseline model during the development of ML, users can simply retain the automatic benchmark as their production model. Beyond benchmark-specific functionalities, these objects behave also as normal scikit-learn models and therefore can be applied as any normal model would.

Benchmarks can be used as a test to see which regressors, if any, differentiate the most between two or more groups, for example to see if one group of samples has out-of-domain data (as proposed by Howard and Gugger (2020a)). The test proceed as follows: a random forest classifier is trained on the dataset, with group identifiers serving as the target variable. The forest's calculated regressor importance

can then uncover what are the variables that differentiate between groups. This test can be generalised to identify which regressors vary substantially between two or more groups.

3.2 Custom automatic benchmarks

In spite of the empirical qualities of random forests, no single algorithm could plausibly cover all potential use cases. For example, random forests could potentially not perform as well as other established algorithms if the economic or financial data at hand contains multimodal data, ie images, videos, texts and other such data. These data types are increasingly relevant in economics research. Some examples of this growing literature on non-traditional data types include Gentzkow, Kelly, and Taddy (2019) and Li et al. (2020) for text and Yeh et al. (2020) and Bajari et al. (2023) for images. In addition, random forests cannot extrapolate beyond the range of the training data that was fed to them. And there are some empirical settings in which gradient boosting trees are more used than random forests and other ML methods (Brotcke (2022)). Similarly, Gorishniy et al. (2021) show that neural networks can achieve similar, and in some cases better, performance than tree-based methods in some cases. And Taylor and Letham (2018) describe Facebook's own tool (now open sourced) for automatic forecasting, without relying on random forests.

So random forests might not be the first choice for all cases, even though they are expected to work well in a wide array of situations. In addition to the off-the-shelf implementations described above, gingado offers two possibilities for users to set up their own benchmark models. The most straightforward one is to simply pass as argument a model object to the benchmark's method set_benchmark. This will cause the benchmark object to put this new model in place of the previous benchmark.

The second object involves the creation of a new benchmark object class altogether. gingado ships with a base class, ggdBenchmark, that contains all the necessary features for a benchmarker object. Users that wish to create their own benchmark models can sub-class from ggdBenchmark and implement the desired funcionalities. The user's benchmark will then work in the same way as the original gingado benchmark models. This user-created benchmark can include any algorithm or combination of algorithms, as long as the API is maintained.

4. Real and synthetic datasets

gingado aims to provide users with an easy way to load real datasets that can be used as benchmarks in research, along with functions that allow users to create synthetic datasets from a wide range of data generating processes.

4.1 Real datasets

Economics and finance research often relies on benchmark datasets used in canonical papers to explore new insights derived from the original work or to evaluation and question the original findings. Prominent (but not exhaustive) examples include the Angrist Data Archive; the data used by Giannone, Lenza and Primiceri (2021) to test ML models; the macro-history datasets of Jordà, Schularick, and Taylor (2017), Jordà

et al. (2019) and Jordà et al. (2021); the technology adoption (CHAT) dataset of Comin and Hobijn (2009); and the datasets used in synthetic control studies by Abadie and Gardeazabal (2003), Abadie, Diamond, and Hainmueller (2010) and Abadie, Diamond, and Hainmueller (2015); and others. Some of these benchmark data may be useful for building and testing ML algorithms.

In many cases, these datasets are shared in a way that makes them convenient to use with almost no manipulation. But even in these cases, these datasets are formatted using different platforms (Stata's DTA file, or in CSV/Excel files, etc). What gingado aims is to make available selected benchmark datasets in a standardised format that can be readily used by economists. At this point, the only dataset loaded this way in gingado is the one from Barro and Lee (1994); Listing 4 illustrates its use. After importing the load_BarroLee_1994 function in the first line, the user then attributes the result of calling this function to two datasets, X and y, which can then be used as normal for the training of ML models. The goal is for other datasets to have a similar structure, where their usage can be as simple as importing the appropriate function and calling it once. Naturally, the original source of the all data used should be cited and acknowledged accordingly by the end user.

Importantly, gingado does not aim to restrict this section only to datasets that support well-cited papers. Users that want to propose other datasets, including from their own work, are welcome to do so. In any case, the data must be used in a published academic paper.

Loading data from Barro and Lee (1994)

Python environment — Listing 4

```
from gingado.datasets import load_BarroLee_1994()
X, y = load_BarroLee_1994()
```

Sources: Author.

4.2 Simulated datasets

In many cases, researchers testing new econometric estimators use simulated data, created under "lab-like" conditions with a known data generating process. Such datasets enable the user to test whether proposed estimators really work as intended for a dataset with given characteristics, and can be especially helpful for causal estimands (Imbens and Rubin (2015)). The possibility of simulating a datasets of varying lengths - on the time and the cross-sectional dimensions - also facilitate the testing of asymptotics. gingado's make_causal_effect function offers users the possibility to simulate non-trivial data, including with non-linear interactions and a rich set of treatment-related variables.

More specifically, make_causal_effect allows users to creates a dataset $\{\mathrm{y_i}, \mathrm{X_i}, \mathrm{D_i}\}$ of outcome variable, covariates and treatment vectors respectively, with the number of samples $i = (1, \dots, n)$ of the dataset, which can be interpreted as peer units or as time periods in a panel data, and $k = (1, \dots, M)$ number of features. Users are able to specify the functional form by adjusting the following characteristics:

- $y_i \mid X_i$, the pre-treatment outcome. For the untreated units, this corresponds to the observations of y_i; for the treated units, this is the portion of y_i before adding

the treatment effect. The pre-treatment outcome depends on the covariates X_i and on a constant. It might also have a random component[6]

- $W_i = p(D_i \neq 0 \,|X_i)$, i's propensity of being treated, ie having a treatment that is different than zero.[7] The propensity can be a function that depends on $\$X_i$, either deterministically or with a random component. Alternatively, it can also be set with a scalar, in which case it is uniform across all samples, or with a random assignment to each sample according to a parameterised or empirical random distribution; in both cases the propensity simplifies to $p(W_i = 1 \,|X_i) = p(W_i = 1)$.
- $D_i \neq 0 \,|\, W_i$, the actual treatment assignment. The default value is purely a function of the treatment propensity through a binomial distribution: $\$p(D_i \neq 0 \,|W_i) = B(1, W_i)\$$. This is probably the most relevant use case. However, there are instances where researchers might want to explore a more complex treatment assignment relationship. For example, treatment can be rationed and only applied to the ψ samples with the highest W_i, or applied with probability $B(1, W_i)$ but also subject to this rationing. In these cases, the treatment assignment would also be a function of ψ, in addition to W_i. For example, the case where only the ψ most propense units would be treated can be defined as: $1(D_i \neq 0) = B(1, W_i) \times 1(B(1, W_i) \in \text{top}(W_i, \psi))$, where $\text{top}(\cdot, n)$ is the function that orders the set and returns the highest n observations.
- D_i, the treatment value. The treatment can be set to 1 in the most simple of cases, but the treatment magnitude (and sign) can also vary as a function of covariates X_i, for $D_i \,|\, X_i$. A random variation in the sample-specific treatment can also be introduced.
- $\tau_i |\, D_i$, the treatment effects on y_i for each treated sample. This represents the difference between the actual observation y_i and the pre-treatment outcome y_i for treated samples. In the most simple cases, it may be a one-to-one mapping to the treatment value D_i. But, it may also vary by sample according to covariates X_i.

All dependencies on covariates above can include non-linearities such as minimum or maximum comparisons, various types of interactions, exponentials, etc. In short, anything that can be coded using a NumPy array and respects the necessary constraints of each argument (for example, the treatment propensity must always be a number between 0 and 1). It is also important to note that a more complex treatment chain (propensity to assignment to value to effect) can depend on covariates in a different way at each step. Hopefully this flexibility in creating datasets with causal effects can provide researchers with ways to compare different ways to correct for interferences in the estimation of different potential outcomes.

[6] All random components in gingado code can be made reproduceable with the use of the same random seed.

[7] The treatment value itself can be set by the user, and therefore is not necessarily a binary dummy.

5. Model documentation

Stakeholders often require some level of documentation of the ML models. From simple descriptions of the model to standardised model reports to fully-fledged evaluation of models, gingado provides a way for models to be more easily documented. The basic idea is that a *documentation template* outlines the specific items to be documented, and various methods allow the user to interact with this template. This template can be seen at any time by the user with the method show_template. A gingado documenter object also specifies which of those items are to be filled in automatically (eg, model description can be parsed from the ML algorithm itself), and how exactly this should be done.

After a documenter is instantiated, it can read information from an existing model as well. Listing 5 demonstrates how straightforward it is to automatically read information from a model, and then see what are the questions from the documentation template that were not answered automatically by the ModelCard object, even when the ML model itself is not a gingado object.[8] The first two rows import the necessary objects: a gingado.ModelCard class and the keras ML library. The following chunk establishes the structure of a neural network comprising of two dense layers (each with 16 nodes) followed by an end layer that predicts the probability in a classification model. This neural network is then fit using training data frames X_train and y_train assumed in Listing 5 to already exist, as before. Now, an instance of ModelCard object, called model_doc_keras is created, and it then reads the information from the keras classification ML model that was created and trained right before. Finally, in the last line the code presents to the user what are the questions, or fields, in the model documentation that are still open because they were not read automatically. This nudges the user to consider answering these questions (and thus documenting their model in an easy and recordable way).

Example of ModelCard reading information from an existing neural network model

Python environment — Listing 5

```
from gingado.model_documentation import ModelCard
import keras_core as keras

keras_clf = keras.Sequential()
keras_clf.add(keras.layers.Dense(
    16,
    activation='relu',
    input_shape=(20,)
    ))
keras_clf.add(keras.layers.Dense(16, activation='relu'))
```

[8] Currently the base documenter ggdModelDocumentation, from which all documenters in this library derive, can read models created using gingado, scikit-learn and keras. Support for automatically reading information from models built with other libraries such as PyTorch is under development.

```
keras_clf.add(keras.layers.Dense(1, activation='sigmoid'))
keras_clf.compile(optimizer='sgd', loss='binary_crossentropy')

keras_clf.fit(X_train, y_train, batch_size=10, epochs=10)

model_doc_keras = ModelCard()
model_doc_keras.read_model(keras_clf)
model_doc_keras.open_questions()
```

Sources: Author.

At any time, the user can view the current state of the document with the method show_json, and save or read it to file with save_json and read_json respectively. Additional information items that were not included automatically are filled with fill_info (the user can also override automatic entries). And the remaining items from the template that are not yet filled are shown to the user with the method open_questions. These methods (and others, not shown for brevity) aim to provide the user with a more direct, hands-on approach to documenting the model, compared to a more traditional setting where a separate document (ie, an MS Word file) need to be written and maintained. Allowing users to document their models from inside their ML development environment will help embed the documentation process as another step of the model development. Another advantage of gingado's approach is that it is much easier to keep the documentation aligned with the current version of the model. This is particularly important in the settings where the user expects to iterate over different specifications until a suitable model is achieved.

The model documentation is stored as a JSON file, a flexible format that is easy for machines to read, and that can quickly be transformed into objects humans can read more easily, too. gingado uses JSON files because they are a common language to serialise this type of structured information that works across platforms (ie, a JSON file works in Windows machines the same way it works in MacOS, Linux or any other system). Users that want to automatically produce documents in other formats (eg, PDF files) can do so by using these JSON files with other libraries of their choice. And in settings where multiple models are developed and in production, JSON files are a more streamlined format to offer information on each model to comparison scripts.

gingado includes two ready-to-use documenter objects, ModelCard and ForecastCard. The ModelCard documentation template is based on the work of Mitchell et al. (2018), which I believe strikes a good balance between being a general documentation template while also prompting the user to answer questions about the model that are relevant from a broader stakeholder perspective.ForecastCard is a version of ModelCard with questions that are targeted for forecasting tasks. Similar to how users can create their own class of benchmark models, gingado enables users to create their own custom documenters, from the base class ggdModelDocumentation. This allows specific documentation templates to be used in a more automatic way, and can be of particular importance in the context of organisations using ML, since they might have their own documentation preferences. Users can also benefit from the machinery underlying the ggdModelDocumentation base class to create dataset documentation (eg, à la Gebru et al. (2021)).

The template for the ForecastCard can provide an example of the type of information a documenter could either acquire automatically from reading the model object and from asking the economist; note that its language is only slightly adapted from Mitchell et al. (2018) with some fields directly reproducing a model card after those authors:

- Model details (basic information about the model)
  - Variable(s) being forecasted or nowcasted
  - Jurisdiction(s) of the variable being forecasted or nowcasted
  - Person or organisation developing the model
  - Model date
  - Model version
  - Model type
  - Description of the pipeline steps being used
  - Information about training algorithms, parameters, fairness constraints or other applied approaches, and features
  - Information about the econometric model or technique
  - Paper or other resource for more information
  - Citation details
  - License
  - Where to send questions or comments about the model
- Intended use (use cases that were envisioned during development)
  - Primary intended uses
  - Primary intended users
  - Out-of-scope use cases
- Metrics (metrics should be chosen to reflect potential real world impacts of the model)
  - Model performance measures
  - How are the evaluation metrics calculated? Include information on the cross-validation approach, if used
- Data (details on the dataset(s) used for the training and evaluation of the model
  - Datasets
  - Preprocessing steps
  - Cut-off date that separates training from evaluation data
- Ethical considerations (considerations that went into model development, surfacing ethical challenges and solutions to stakeholders. Ethical analysis does not always lead to precise solutions, but the process of ethical contemplation is worthwhile to inform on responsible practices and next steps in future work)
  - Does the model use any sensitive data?

  - What risks may be present in model usage? Try to identify the potential recipients, likelihood, and magnitude of harms. If these cannot be determined, it can be noted that they were considered but remain unknown
  - Are there any known model use cases that are especially fraught?
  - If possible, this section should also include any additional ethical considerations that went into model development, for example, review by an external board, or testing with a specific community.
- Any other caveats or recommendations (additional concerns that were not covered in the previous sections)
  - For example, did the results suggest any further testing? Were there any relevant groups that were not represented in the evaluation dataset?
  - Are there additional recommendations for model use? What are the ideal characteristics of an evaluation dataset for this model?

While most economists trained in traditional econometric techniques using time series quantitative data for forecasting might find some of these fields "over-kill", the objective of this documenter is to be a reasonably simple template for models that might, who knows, forecast economically variables by, say, also including textual data ou some other form of non-traditional data.

5.1 Ethical issues in economics and finance for ML models

One of the reasons why gingado facilitates model documentation is to promote a greater role for ethical considerations as part of the development of ML models in economics and finance: if the model documentation becomes part of development workflow and certain parts of the documentation are automated, then users are presumably more likely to consider these issues at development time, not *ex post*. The importance of ethical considerations in finance applications of ML is underscored by the ability of ML to drive results in economics and finance: for example, Frost et al. (2019) show evidence that ML models can outperform traditional credit bureau data in predicting loan default rate in Argentina, which illustrates the strong incentives for investment in ML models by lenders. Other evidence of the real-life impact of ML applications might materially impact stakeholder outcomes is studied by Fuster et al. (2022), in the context of US mortgage lending. These authors, who also document evidence of ML outperforming traditional models, find that greater use of ML would lead to relatively higher estimated propensities of default for Black and Hispanic borrowers, even when race is not used as a feature in the models.

More broadly, as discussed by Rambachan et al. (2020), differences in predictions between groups that might be attributable to algorithmic bias can be seen as a combination of basal differences observed in society (and itself possibly the result of a societal bias), and of measurement error and estimation error differences. Of course, only the last two can be addressed by better developed ML models. Still, simply bringing the existence of this bias to light as done during the documentation process might be helpful in driving economists to make models that address this issue at least partially (Cowgill et al. (2020)). Further to that, models that are likely to result in algorithmic bias even from predominantly underlying societal causes should ideally make it clear for users that this could be the case, so that users can exercise due discretion in whether and how to deploy these models. Another strategy that could also be documented with gingado is to choose different samples of the data based

on subpopulations in which the outcome is perceived or known to be less biased, as suggested by Ludwig and Mullainathan (2021). In any case, it is plausible to expect that real-life models where these issues loom large should include documentation describing the strategy taken by model developers.

There seems to be a wide awareness of the implications from biased datasets feeding into complex, black-box ML models in economics and finance.[9] They are illustrated for example by Brotcke (2022), who present a practitioner view on using ML while seeking to originate loans without bias, by the law enforcement examples of ML gone awry discussed in Ludwig and Mullainathan (2021) and by Doerr, Gambacorta, and Garralda (2021), who describe central banks' concerns with fairness implications from greater use of ML. In the broader language domain, Bender et al. (2021) highlight ethical and societal issues stemming from the incredibly large size of datasets used to train large language models (LLMs).

But the ethical implications of ML do not stem just from the datasets used for training. It is likely that similar issues are important in a variety of uses, in varying degrees. For example, Bender et al. (2021) also discuss model-related aspects: eg, the environmental impact of the large-size architecture of these models, and the risks originating from the higher (apparent) fluency of these models. Mullainathan and Obermeyer (2017) and Thomas and Uminsky (2022) discuss pitfalls and risks from the metrics chosen during ML development. And importantly in the case of economics, Ludwig and Mullainathan (2021) pinpoint the poor experience with ML in the judicial system to a case of faulty development of models, including due statistical consideration to the fact that modeled outcomed are in many cases only a subset of the necessary information, and that decision is taken ultimately by a human decisionmaker. In addition, properly informing the user on algorithmic design choices and recommended and unrecommended use cases might prevent situations where the model is designed to be unbiased, but its deployment is botched due to not considering that the model would work best if delivered equally to all subpopulations of interest, as shown by Lambrecht and Tucker (2019) in the case of an ML-delivered ad for careers in sciences, technology, engineering and maths (STEM) designed to be gender neutral but that was more widely seen by men due to the higher ad costs for women. Therefore, promoting opportunities for economists developing and deploying ML models to think about these implications from the complete model pipeline, from data to application to usage, seems more than warranted, in line with Cowgill and Tucker (2019).

Consideration of these and other ethical issues might be made easier by facilitating model documentation, which nudges users to explicitly document (and thus consider) features of the data and model that go beyond purely model architecture features, which are important but can be read automatically by software. In fact, I hope that the mere availability of model documentation functionalities might act as a reminder that this is an important step (Cowgill et al. (2020)). At the same time, automatically covering model information in the documentation opens up space for the user to reflect on the open human-answerable questions, many of which include issues around ethics. In other words, can model documentation address all issues in ML ethics in economics? Definitely no. But the hope is that enabling and

[9] A related discussion where ethics and fairness in ML has made strides but needs more progress is related to explainable artificial intelligence (XAI) (for example, Barredo Arrieta et al. (2020)).

facilitating model documentation might help users take steps in that direction, both in academic as well as in practitioner settings.

6. What to expect next?

The vision for gingado is for it to become a collection of tools that can help economists deploy ML in various use cases in research or practitioner work. In this sense, development of new tools to be added in gingado is guided by two considerations. First, what are the pain points of the ML workflow for economists that can be tackled ergonomically? Second, what are the areas of ML that can benefit economics use cases? Using these considerations as backdrop, four areas of active development as of writing of this paper are:

- new canonical datasets to help new users get up to speed with initial models, fomenting the build-up of AI skillsets for beginner-level users or simply being used as benchmarks;
- clustering functionalities that automatically divide a population of entities into clusters, and retain only those that are in the same cluster as a specified entity of interest. This can be used to find (and retain only) related entities to unit(s) of interest in multidimensional settings, providing a useful selection of control units for example, for matching applications (Imbens and Wooldridge (2009)) or in the estimation of causal effects using synthetic controls (Abadie and Gardeazabal (2003), Abadie (2021)) in a more data-driven way, as proposed by Araujo et al. (2023).
- causal ML, implementing Python versions of algorithms that have been designed statistically to allow for causal inference. Examples that might be implemented include the generalised random forest of Athey, Tibshirani, and Wager (2019), the ML-based version of synthetic controls of Araujo et al. (2023), the double/debiased ML of Chernozhukov, Chetverikov, et al. (2018) and other algorithms.
- functionalities to use and adapt LLMs (eg, OpenAI (2023), Touvron et al. (2023)) in settings of economic interest.

7. Conclusion

gingado is a free, open source ML library focused on use cases in economics and finance - in both research and practice. Its compatibility with widely used ML frameworks, most notably scikit-learn, allows it to leverage on the wide familiarity with these tools and complement existing user codebases. And its flexible approach simultaneously affords users the ability to advance ML model development with few steps, while enabling users to tweak all tools to meet their goals, including adjusting models and their outcomes to better suit econometric use cases when necessary (Athey and Imbens (2019)). The toolset provided by gingado was first created for my own use as a practitioner of ML in economics, and I will continue developing it over time to incorporate funcionalities that can be of most value-added to researchers and practitioners in economics and finance. Because the code is open, also the broader

public can propose improvements and even new funcitonalities, either by directly suggesting it or by proposing code that implements these ideas. At the same time, I hope that gingado can contribute to facilitate greater use of ML in economics and finance, while promoting good modelling practices including a greater role for model documentation and ethical considerations.

Of course, both research and empirical applications are already benefiting from more intense usage of ML, including in policy organisations (FSB (2017), Araujo et al. (2023), Frost et al. (2019), Doerr, Gambacorta, and Serena (2021))). And recent breakthroughs in the field (such as GPT-4, OpenAI (2023)) are likely to further promote this, as exemplified by Korinek (2023). In this context, gingado can help new and more experienced economists familiarise themselves with ML practice in an ergonomic way, by removing some of the complexity in getting such models set up, trained, and documented. The particular areas focused by gingado so far are data augmentation, automatic benchmark models, real and simulated datasets and model documentation.

Adding more data to one's dataset can often be cumbersome from an operational perspective, and especially so when multiple sources are involved. This is of course much harder when the model is used in a production setting, instead of a one-off analysis. gingado addresses this by augmenting the user dataset through an object that fits nicely in standard ML pipelines. This also allows the user to test whether or not adding more data actually improves the model (according to any criteria defined by the user). In addition, gingado's data augmentation method focuses on ensuring that the data provided to users is from trusted sources. When more data augmentation objects are added to the gingado codebase, the reliability of data sources will be a key criterion.

Automatic benchmark models are not new. Howard and Gugger (2020b) and Taylor and Letham (2018) for example enable users to quickly set up models with a reasonably good performance for the vast majority of use cases in their respective domains. This is also followed for tabular data by Apple's CreateML framework that underpins numerous ML applications in Apple devices. gingado builds on that insight and provides users with additional functionalities that to my knowledge are novel. Namely, it offers a way to conveniently compare candidate models (and their ensemble) and pick the best one as the new benchmark. It also offers the automatic documentation of the benchmark model, and the ability to create one's own benchmark from a base class that that ensures users' customised benchmark models would be compatible with the other functionalities.

These models, of course, need to be trained on data. gingado provides the user with real and simulated datasets. The former currently contains one dataset, used by Barro and Lee (1994), but others will be included over time with the goal of forming a portfolio of academic benchmark data representing various areas of economics and finance studies. And the functionality to simulate data can be used to generate a wide range of datasets, with rich treatment chains and non-linear dependences of the outcome variable on the covariates. These can be especially useful in the context of causal ML using the potential outcomes framework (Imbens and Rubin (2015)).

And finally, there is model documentation. Mullainathan and Spiess (2017) mention the risk that ML models in economics and finance are applied naïvely or have their outputs misinterpreted. This risk increases with greater deployment of ML in important aspects of daily life, such as banking. But the risk probably also grows as the technical preconditions for ML, such as greater availability of higher-dimensional

datasets and of compute power, facilitate model development by more people. As the popularity of ML models amongst economists grows, my goal with gingado is to contribute a small part to embed model documentation in the process of model development, automating some of the questions to afford the humans in control the opportunity to properly document their models and pay due consideration to ethical issues arising in each particular situation, hopefully facilitating research in the field and leading to better AI models developed and deployed in practice.

References

Abadie, Alberto (2021): "Using Synthetic Controls: Feasibility, Data Requirements, and Methodological Aspects", *Journal of Economic Literature*, v. 59, n. 2, p. 391–425.

Abadie, Alberto, Alexis Diamond, and Jens Hainmueller (2010): "Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California's Tobacco Control Program", *Journal of the American Statistical Association*, vol 105, n. 490, p. 493–505.

Abadie, Alberto, Alexis Diamond, and Jens Hainmueller (2015): "Comparative Politics and the Synthetic Control Method", *American Journal of Political Science*, vol 59, n. 2, p. 495–510.

Abadie, Alberto and Javier Gardeazabal (2003): "The Economic Costs of Conflict: A Case Study of the Basque Country", *American Economic Review*, vol 93, n. 1, p. 113–32.

Araujo, Douglas, Giuseppe Bruno, Juri Marcucci, Rafael Schmidt, and Bruno Tissot (2023): "Machine Learning Applications in Central Banking", *Journal of AI, Robotics and Workplace Automation*, vol 2, n. 3.

Athey, Susan and Guido W. Imbens (2019): "Machine Learning Methods That Economists Should Know About", *Annual Review of Economics*, vol 11, n. 1, p. 685–725.

Athey, Susan, Julie Tibshirani, and Stefan Wager (2019): "Generalized Random Forests".

Bajari, Patrick, Zhihao Cen, Victor Chernozhukov, Manoj Manukonda, Suhas Vijaykunar, Jin Wang, Ramon Huerta, et al. (2023): "Hedonic Prices and Quality Adjusted Price Indices Powered by AI", *arXiv Preprint* arXiv:2305.00044.

Barredo Arrieta, Alejandro, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, et al. (2020): "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities, and Challenges Toward Responsible AI", *Information Fusion*, vol 58, p. 82–115.

Barro, Robert J., and Jong-Wha Lee. 1994. "Sources of Economic Growth." *Carnegie-Rochester Conference Series on Public Policy* 40: 1–46. https://doi.org/https://doi.org/10.1016/0167-2231(94)90002-7.

Bender, Emily M., Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, 610–23. FAccT '21*. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3442188.3445922.

FSB, Financial Stability Board. 2017. Artificial Intelligence and Machine Learning in Financial Services: Market Developments and Financial Stability Implications.

Breiman, Leo. 1996. "Bagging Predictors." *Machine Learning* 24 (2): 123–40.

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32.

Brotcke, Liming. 2022. "Time to Assess Bias in Machine Learning Models for Credit Decisions." *Journal of Risk and Financial Management* 15 (4). https://doi.org/10.3390/jrfm15040165.

Chakraborty, Chiranjit, and Andreas Joseph. 2017. "Machine Learning at Central Banks."

Chernozhukov, Victor, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. 2018. "Double/debiased machine learning for treatment and structural parameters." *The Econometrics Journal* 21 (1): C1–68. https://doi.org/10.1111/ectj.12097.

Chernozhukov, Victor, Mert Demirer, Esther Duflo, and Ivan Fernandez-Val. 2018. "Generic Machine Learning Inference on Heterogeneous Treatment Effects in Randomized Experiments, with an Application to Immunization in India." *National Bureau of Economic Research*.

Comin, Diego A, and Bart Hobijn. 2009. "The CHAT Dataset." Working Paper 15319. Working Paper Series. National Bureau of Economic Research. https://doi.org/10.3386/w15319.

Cowgill, Bo, Fabrizio Dell'Acqua, Samuel Deng, Daniel Hsu, Nakul Verma, and Augustin Chaintreau. 2020. "Biased Programmers? Or Biased Data? A Field Experiment in Operationalizing AI Ethics." In Proceedings of the 21st ACM Conference on Economics and Computation, 679–81.

Cowgill, Bo, and Catherine E Tucker. 2019. "Economics, Fairness and Algorithmic Bias." Preparation for: Journal of Economic Perspectives.

Doerr, Sebastian, Leonardo Gambacorta, and José Maria Serena Garralda. 2021. "Big Data and Machine Learning in Central Banking." BIS Working Papers, no. 930.

Doerr, Sebastian, Leonardo Gambacorta, and Jose Maria Serena. 2021. "How Do Central Banks Use Big Data and Machine Learning." In The European Money and Finance Forum, 37:1–6.

Duarte, Victor. 2018. "Machine Learning for Continuous-Time Economics." Available at SSRN 3012602.

Fernández-Villaverde, Jesús, Samuel Hurtado, and Galo Nuno. 2019. "Financial Frictions and the Wealth Distribution." National Bureau of Economic Research.

Ferreira, Leonardo N et al. 2021. "Forecasting with VAR-teXt and DFM-teXt Models: Exploring the Predictive Power of Central Bank Communication."

Frost, Jon, Leonardo Gambacorta, Yi Huang, Hyun Song Shin, and Pablo Zbinden. 2019. "BigTech and the Changing Structure of Financial Intermediation." Economic Policy 34 (100): 761–99.

Fuster, Andreas, Paul Goldsmith-Pinkham, Tarun Ramadorai, and Ansgar Walther. 2022. "Predictably Unequal? The Effects of Machine Learning on Credit Markets." The Journal of Finance 77 (1): 5–47. https://doi.org/https://doi.org/10.1111/jofi.13090.

Gebru, Timnit, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. 2021. "Datasheets for Datasets." Communications of the ACM 64 (12): 86–92.

Gentzkow, Matthew, Bryan Kelly, and Matt Taddy. 2019. "Text as Data." Journal of Economic Literature 57 (3): 535–74.

Géron, Aurélien. 2019. Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition. Sebastopol, CA: O'Reilly Media.

Giannone, Domenico, Michele Lenza, and Giorgio E Primiceri. 2021. "Economic Predictions with Big Data: The Illusion of Sparsity." Econometrica 89 (5): 2409–37.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. MIT Press.

Gorishniy, Yury, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. "Revisiting Deep Learning Models for Tabular Data." https://proceedings.neurips.cc/paper/2021/hash/9d86d83f925f2149e9edb0ac3b49229c-Abstract.html.

Howard, Jeremy, and Sylvain Gugger. 2020a. Deep Learning for Coders with Fastai and Pytorch: AI Applications Without a PhD. O'Reilly Media, Incorporated.

Howard, Jeremy, and Sylvain Gugger. 2020b. "Fastai: A Layered API for Deep Learning." Information 11 (2). https://doi.org/10.3390/info11020108.

Hünermund, Paul, Beyers Louw, and Itamar Caspi. 2023. Journal of Causal Inference 11 (1): 20220078. https://doi.org/doi:10.1515/jci-2022-0078.

Imbens, Guido W, and Donald B Rubin. 2015. Causal Inference in Statistics, Social, and Biomedical Sciences. Cambridge University Press.

Imbens, Guido W., and Jeffrey M. Wooldridge. 2009. "Recent Developments in the Econometrics of Program Evaluation." Journal of Economic Literature 47 (1): 5–86. https://doi.org/10.1257/jel.47.1.5.

Jordà, Òscar, Katharina Knoll, Dmitry Kuvshinov, Moritz Schularick, and Alan M Taylor. 2019. "The Rate of Return on Everything, 1870–2015." The Quarterly Journal of Economics 134 (3): 1225–98.

Jordà, Òscar, Björn Richter, Moritz Schularick, and Alan M Taylor. 2021. "Bank Capital Redux: Solvency, Liquidity, and Crisis." The Review of Economic Studies 88 (1): 260–86.

Jordà, Òscar, Moritz Schularick, and Alan M Taylor. 2017. "Macrofinancial History and the New Business Cycle Facts." NBER Macroeconomics Annual 31 (1): 213–63.

Kleinberg, Jon, Jens Ludwig, Sendhil Mullainathan, and Ziad Obermeyer. 2015. "Prediction Policy Problems." American Economic Review 105 (5): 491–95. https://doi.org/10.1257/aer.p20151023.

Kohlscheen, Emanuel. 2021. "What Does Machine Learning Say about the Drivers of Inflation?" Available at SSRN 3949352.

Korinek, Anton. 2023. "Language Models and Cognitive Automation for Economic Research." Working Paper 30957. Working Paper Series. National Bureau of Economic Research. https://doi.org/10.3386/w30957.

Lambrecht, Anja, and Catherine Tucker. 2019. "Algorithmic Bias? An Empirical Study of Apparent Gender-Based Discrimination in the Display of STEM Career Ads." Management Science 65 (7): 2966–81. https://doi.org/10.1287/mnsc.2018.3093.

Li, Kai, Feng Mai, Rui Shen, and Xinyan Yan. 2020. "Measuring Corporate Culture Using Machine Learning." The Review of Financial Studies 34 (7): 3265–315. https://doi.org/10.1093/rfs/hhaa079.

Ludwig, Jens, and Sendhil Mullainathan. 2021. "Fragile Algorithms and Fallible Decision-Makers: Lessons from the Justice System." Journal of Economic Perspectives 35 (4): 71–96. https://doi.org/10.1257/jep.35.4.71.

Maliar, Lilia, Serguei Maliar, and Pablo Winant. 2021. "Deep Learning for Solving Dynamic Economic Models." Journal of Monetary Economics 122: 76–101.

Mitchell, Margaret, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2018. "Model Cards for Model Reporting." CoRR abs/1810.03993. http://arxiv.org/abs/1810.03993.

Mullainathan, Sendhil, and Ziad Obermeyer. 2017. "Does Machine Learning Automate Moral Hazard and Error?" American Economic Review 107 (5): 476–80. https://doi.org/10.1257/aer.p20171084.

Mullainathan, Sendhil, and Jann Spiess. 2017. "Machine Learning: An Applied Econometric Approach." Journal of Economic Perspectives 31 (2): 87–106. https://doi.org/10.1257/jep.31.2.87.

OpenAI. 2023. "GPT-4 Technical Report." https://arxiv.org/abs/2303.08774.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." Journal of Machine Learning Research 12: 2825–30.

Rambachan, Ashesh, Jon Kleinberg, Jens Ludwig, and Sendhil Mullainathan. 2020. "An Economic Perspective on Algorithmic Fairness." AEA Papers and Proceedings 110 (May): 91–95. https://doi.org/10.1257/pandp.20201036.

Rossi, Barbara. 2013. "Exchange Rate Predictability." Journal of Economic Literature 51 (4): 1063–1119.

Stierholz, Katrina. 2014. "FRED®, the St. Louis Fed's force of data." Review 96 (2): 195–98. https://ideas.repec.org/a/fip/fedlrv/00023.html.

Taylor, Sean J, and Benjamin Letham. 2018. "Forecasting at Scale." The American Statistician 72 (1): 37–45.

Thomas, Rachel L, and David Uminsky. 2022. "Reliance on Metrics Is a Fundamental Challenge for AI." Patterns 3 (5): 100476.

Touvron, Hugo, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, et al. 2023. "Llama 2: Open Foundation and Fine-Tuned Chat Models." arXiv Preprint arXiv:2307.09288.

Ushey, Kevin, JJ Allaire, and Yuan Tang. 2022. Reticulate: Interface to 'Python'.

Varian, Hal R. 2014. "Big Data: New Tricks for Econometrics." Journal of Economic Perspectives 28 (2): 3–28.

Yeh, Christopher, Anthony Perez, Anne Driscoll, George Azzari, Zhongyi Tang, David Lobell, Stefano Ermon, and Marshall Burke. 2020. "Using Publicly Available Satellite Imagery and Deep Learning to Understand Economic Well-Being in Africa." Nature 11. https://doi.org/https://doi.org/10.1038/s41467-020-16185-w.

gingado: a machine learning 🤖 library focused on economics and finance

/ʒĩ.ˈga.du/

IFC-Bank of Italy workshop on “Data science in central banking”, Part 2

What is *gingado*?

- Open-source machine learning library written in Python (under development 🛠️)
- Objective: broaden the accessibility of state-of-the-art models to a wide range of practitioners in economics and finance
- Main features:

1. Automatic benchmark models
2. Automatic data augmentation
3. Automatic documentation

… all of that with a simple API

Automatic benchmark models

- Once the model and the dataset are defined, *gingado* automatically trains a benchmark model (unless asked not to train it)
- Benchmark models are useful to compare results from user attempts
- Worse case scenario, if all your attempts fail at beating the benchmark, at least you have a reasonable model

Automatic data augmentation

- “Data augmentation” means to append more data to your dataset. This is known to generally improve the performance of ML models.
- For example, in ML models working with image, data augmentation involves flipping, cutting, zooming in or out, etc.
- In economic/financial datasets, augmentation involves adding other information related to the observations in the dataset
- For example, if the dataset is a country-level panel data, data augmentation would add a lot of publicly-available data for the countries in the dataset
- New data sourced using sdmx and other APIs from official sources

Automatic data augmentation

Ensuring the data augmentation works

1. user defines the variables of interest for augmentation: time + geographies
2. *gingado* fetches data from BIS, IMF, ECB, Eurostat, & other sources
3. a new benchmark model is run with *all* data and it is compared to the original
4. all of the augmented data is kept if performance improves
5. if performance does not improve, only the variables that have low correlation to all of the variables in the origination dataset are kept

Automatic model documentation

- Model card inspired by Mitchell et al (2019)
- “Meta-data” about the model
- Transparency about:
  - envisage use contexts
  - how the model was evaluated, etc
- Important in model development, management, audits
- *gingado* uses JSON to store the raw model card info

Model Card

- **Model Details**. Basic information about the model.
  - Person or organization developing model
  - Model date
  - Model version
  - Model type
  - Information about training algorithms, parameters, fairness constraints or other applied approaches, and features
  - Paper or other resource for more information
  - Citation details
  - License
  - Where to send questions or comments about the model
- **Intended Use**. Use cases that were envisioned during development.
  - Primary intended uses
  - Primary intended users
  - Out-of-scope use cases
- **Factors**. Factors could include demographic or phenotypic groups, environmental conditions, technical attributes, or others listed in Section 4.3.
  - Relevant factors
  - Evaluation factors
- **Metrics**. Metrics should be chosen to reflect potential real-world impacts of the model.
  - Model performance measures
  - Decision thresholds
  - Variation approaches
- **Evaluation Data**. Details on the dataset(s) used for the quantitative analyses in the card.
  - Datasets
  - Motivation
  - Preprocessing
- **Training Data**. May not be possible to provide in practice. When possible, this section should mirror Evaluation Data. If such detail is not possible, minimal allowable information should be provided here, such as details of the distribution over various factors in the training datasets.
- **Quantitative Analyses**
  - Unitary results
  - Intersectional results
- **Ethical Considerations**
- **Caveats and Recommendations**

Source: Mitchell et al (2019)

Backend

- Backend based on *fast.ai* library 
- *gingado* inherits the following characteristics from *fast.ai*:
  - simple yet hackable
  - flexibility to include non-numeric datasets (texts, etc)
  - production-ready results
  - excellent community support for any *fast.ai*-specific issues
- Extensive use of 

as well

If you are interested…

Please use it and share your experience!

- Open up an “issue” on GitHub if you experience any bug
- You may also open up “issues” for feature requests (please be as specific as possible)
- Contributors are welcome! Please get in touch before opening a pull request

https://github.com/dkgaraujo/gingado/

Thank you for your attention!