



gingado: a machine learning library focused on economics and finance

Douglas Araujo, BIS*

Royal Statistical Society, Finance and Economics Special Interest Group meeting, 26 May 2022

* The views expressed here are my own and not those of the Bank for International Settlements

Overview

- *gingado* seeks to facilitate the use of machine learning in economic and finance use cases, while promoting good practices
- Its main contributions are:
 - **Data augmentation** with information from official sources that is relevant for each case
 - **Automatic benchmark** models trained quickly for the dataset at hand
 - **Model documentation** as part of the work flow
- Design principles
 - **Flexibility**: users can use *gingado* out of the box or build custom processes
 - **Compatibility**: *gingado* works well with other widely used libraries in data science / ML
 - **Responsibility**: documentation and ethical considerations are key steps in modelling

Machine learning is increasingly used in economics and finance...

- Academic uses
 - Some recent papers: Fuster et al (2021), Dong et al (2021), Chernozhukov et al (2021)
 - Many others use ML to help build the dataset of interest, estimate the models, or as objects of study
- Practitioners
 - Lenders using ML to screen loan applications
 - Uncover suspicious money laundering transactions
 - Forecasts of prices for trading purposes, other use cases...
- Use cases span classification, regression, clustering, prediction/forecasting, and others

... but it is still cumbersome to set up and train these models adequately

- Increasing availability of official data from multiple sources
 - Few tools enable the collection of this data in a way that is relevant for the user
 - Conversely, more data is not always best and thus it should be more easily testable
- Starting a model from white canvas can be a challenge for many economists
 - It is not straightforward to choose the model(s) to deploy, or to evaluate their results
 - Not many economists are versed in basic software engineering good practices
- Model documentation and ethical considerations may not be in the front seat
 - ... and even when they are, ensuring it is consistent after the many iterations in the model during experimentation often requires effort

Stylised example of current workflow

Exercise: to predict FX rates for the following day

- Obtain the datasets of interest (data vendor, official source, etc)
- Feature engineering:
 - transforming the dataset
 - adding more data (other market variables such as interest rates, stock prices, etc)
- Create a benchmark model (eg, ARIMA)
- Create and evaluating candidate models
- Documenting the model
- Serving the model in a production environment, ensuring the new data goes through the same transformations and data additions as the training data

Data augmentation: rationale for this being a key area in *gingado*

- Thanks to SDMX and to the broader availability of user-friendly data APIs like FRED, querying data from trusted sources during training time has become much easier
 - Accessing data programmatically also allows any transformations and checks for consistency with the original dataset to be done in a reproducible and transparent way
- The above is also true during when the model is in production
 - Some economics/finance models are designed to be run in production, often by other users. Automating the data augmentation helps insulating users from this process
- A programmatic download of relevant additional data from trusted sources (in particular the official sector) promotes good modelling practices

Data augmentation: SDMX

- Statistical Data and Metadata eXchange (SDMX) is an ISO standard comprising:
 - technical standards
 - statistical guidelines, including cross-domain concepts and codelists
 - an IT architecture and tools
- Sponsored by BIS, ECB, EUROSTAT, IMF, OECD, UN, World Bank. Other organisations act as sources of data.
- Informally, each data source publishes multiple **data flows** that users can query according to common definitions across the SDMX standard known as **code lists**.

Data augmentation: example

- The code below demonstrates how *gingado* enables the possibility to augment a dataset with relevant information. This code reads the original dataset's frequency and period, and selects all data from the named sources matching these criteria.

```
● ● ●  
  
# `df` is the original dataset, a time series. Its index is a date.  
df.head()  
  
from gingado.augmentation import AugmentSDMX  
  
augmented_df = AugmentSDMX(sources=['BIS', 'IMF', 'WB']).fit_transform(X=df)
```

- One drawback of this process is that it can be slow depending on the number of sources.

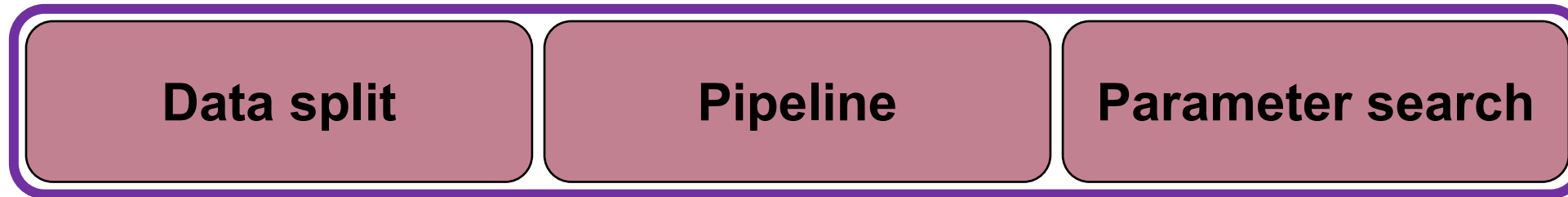
Automatic benchmark: rationale for this being a key area in gingado

- With machine learning, the process of modelling often includes an exploratory phase, but without a goalpost it can be difficult to track improvement
 - This is more frequent in economics and finance use cases where there are few benchmark datasets / tasks to measure models against, unlike say, computer vision
- Beyond that, having a reasonable (even if not optimal) model that takes little time to train can support the iterative nature of the modelling process by offering a “fallback” model in case nothing works satisfactorily
- At the same time, users might have different requirements for benchmark models, so it is important to offer the ability to create customized benchmarks with full compatibility

Automatic benchmark

- Important to note: there is no “best” benchmark that is valid for all uses
 - Still, there might be some techniques that could offer *reasonable performance and transparency* while being trained at a *reasonable speed* to a broad range of uses
- *gingado* offers a generic “ggdBenchmark” object that helps users create their own benchmark model engine.
- Also, an off-the-shelf benchmark model: Regression forests with grid search on parameters.
 - Powerful, versatile algorithm
 - Works well with wide variety of tabular data
 - Don’t require extensive data preparation

Automatic benchmark: scheme



- Depends on whether data is a time series, cross section, or panel data
 - Missing data imputer
 - Random forest
 - Grid search for the random forest parameters
-
- *gingado* benchmarks are deployable by design – if users don't want to pursue any candidate model, they can simply use a “Benchmark” object as the final model.

Automatic benchmark: rationale for random forest

- Does not require extensive data transformation
- Broad application, in particular on tabular datasets
- Relatively fast to train
- Built-in measure of feature importance

Automatic benchmark: rationale for random forest

*“According to our results, model uncertainty is pervasive and **the best prediction is obtained as a weighted average of several models with different sets of regressors**. These findings, in turn, **rationalize the empirical success of** model averaging techniques, and more generally, ensemble machine learning methods such as boosting, bagging, and **random forests**”*

D. Giannone, M. Lenza & G.E. Primiceri, Econometrica Sep 2021, v.89, n.5

Model documentation: rationale for this being a key area in *gingado*

- Documentation should be more automated
 - Each user is likely to have different documentation needs, ranging from:
 - simple human-readable description of the modelling decisions, to
 - more detailed accounts of the modelling process and different evaluation perspectives for auditing purposes
 - At the same time, a user's documentation needs is likely the same for various models from that same user. Therefore, documentation could be (at least partially) automated
- Lowering the bar for documentation will likely lead to more users following that practice
- Since ML is still relatively incipient in economics/finance, offering the possibility to easily jumpstart documentation & ethical questions could help establish this practice more widely

Model documentation: example

- *gingado* includes a generic documentation object so they can build a template according to their own documentation need
- In addition, it comes with an off-the-shelf documentation template, “ModelCard”, inspired by Mitchell et al (2019, cf image)
- While some data points are obtainable automatically by *gingado*, others need to be typed in, offering an opportunity for the user to consider documentation and ethical questions
- Whenever a “Benchmark” object is fit, it is documented, unless the user turns it off

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**

Model documentation: example



```
from gingado.documentation import ModelCard

# ...

# assuming `grid_search` is fitted:
doc = ModelCard(grid_search)

# what information was not obtained automatically?
doc.open_questions( )

# where is the documentation file saved?
print(doc.documentation_path)
```

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**

Technicals

- **Python** package
 - Compatible with (and inspired by) **scikit-learn**, the widely used ML library
 - Currently a pre-alpha release, ie the API may change
 - Also usable in **R**, via {reticulate}
- Open source software
 - Hosted on www.github.com/dkgaraujo/gingado
 - Anyone can contribute – and are welcome to do so!
- Modular
 - Easy to customise data augmentation, benchmark models & documentation
 - Custom objects that seem to apply to other users can also be incorporated in *gingado*

gingado: start to finish example


- This example will illustrate *gingado*'s three main functionalities. First, we import the necessary objects from the library.



```
from gingado.augmentation import AugmentSDMX  
from gingado.benchmark import BenchmarkRandomForest  
from gingado.documentation import ModelCard
```

gingado: start to finish example

- Now we also import the other packages. In this simplified example, the only other package we are using is *scikit-learn*.



```
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor
```

gingado: start to finish example

- Assuming our dataset is already divided appropriately into training and testing datasets, we can already use *gingado* to create and train a benchmark model.

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

- Behind the scenes, this object:
 - Recognises whether this is a regression or a classification task (the user can also choose)
- pipeline {
- Interacts all features with each other, and includes lags if data is a time series
 - Creates a random forest model, conducts a parameter search and retains the best model
 - Documents the model

gingado: start to finish example

- With a benchmark model in hand, we explore whether other models will yield better results. For that, we will use another algorithm in this example (K-Nearest Neighbors) in a pipeline.

```
pipeline = Pipeline([
    ('augmentation', AugmentSDMX()),
    ('scaler', StandardScaler()),
    ('knn', KNeighborsRegressor())
])
```

- Note that one of the steps is the data augmentation provided by *gingado*. After that step, the pipeline scales the values to have mean = 0 and standard deviation = 1.

gingado: start to finish example

- It is possible to search for the best combination of data sources and model parameters:

```
parameters = {  
    'augmentation__sources': (['ECB'], ['ECB', 'BIS', 'IMF', 'WB']),  
    'knn_weight': ('uniform', 'distance')  
}  
  
grid_search = GridSearchCV(pipeline, parameters, n_jobs=-1)  
grid_search.fit(X, y)
```


gingado: start to finish example

- Now that the model with the best combination of parameters is saved in the object “grid_search”, we can compare it against the original benchmark with the following code:

A dark-themed terminal window with three colored window control buttons (red, yellow, green) in the top-left corner. The text `benchmark.compare(grid_search)` is displayed in a light blue monospace font.

```
benchmark.compare(grid_search)
```

- This command evaluates both models (and their ensemble) and retains the best model as the new benchmark.
- Whenever a new model is chosen as the new benchmark (at inception or because it won over the previous benchmark), this could be added to the documentation if the user wants.



Thank you very much for your attention!
Questions / comments / suggestions?

Planned release: Friday, 27 May 2022 at 22:00 CEST

www.github.com/dkgaraujo/gingado