The Journal of Open Source Software

FreqAI: generalizing adaptive modeling for chaotic time-series market forecasts

Robert A. Caulk Ph.D [1,2], **Elin Törnquist Ph.D** [1,2], **Matthias Voppichler**[2], **Andrew R. Lawless**[2], **Ryan McMullan**[2], **Wagner Costa Santos**[1,2], **Timothy C. Pogue**[1,2], **Johan van der Vlugt**[2], **Stefan P. Gehring**[2], **and Pascal Schmidt**[2]

1 Emergent Methods LLC, Arvada Colorado, 80005, USA **2** Freqtrade open source project

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Statement of need

Forecasting chaotic time-series based systems, such as equity/cryptocurrency markets, requires a broad set of tools geared toward testing a wide range of hypotheses. Fortunately, a recent maturation of robust machine learning libraries (e.g. `scikit-learn`), has opened up a wide range of research possibilities. Scientists from a diverse range of fields can now easily prototype their studies on an abundance of established machine learning algorithms. Similarly, these user-friendly libraries enable "citzen scientists" to use their basic Python skills for data-exploration. However, leveraging these machine learning libraries on historical and live chaotic data sources can be logistically difficult and expensive. Additionally, robust data-collection, storage, and handling presents a disparate challenge. `FreqAI` aims to provide a generalized and extensible open-sourced framework geared toward live deployments of adaptive modeling for market forecasting. The `FreqAI` framework is effectively a sandbox for the rich world of open-source machine learning libraries. Inside the `FreqAI` sandbox, users find they can combine a wide variety of third-party libraries to test creative hypotheses on a free live 24/7 chaotic data source - cryptocurrency exchange data.

Summary

`FreqAI` evolved from a desire to test and compare a range of adaptive time-series forecasting methods on chaotic data. Cryptocurrency markets provide a unique data source since they are operational 24/7 and the data is freely available. Luckily, an existing open-source software, `Freqtrade`, had already matured under a range of talented developers to support robust data collection/storage, as well as robust live environmental interactions for standard algorithmic trading. `Freqtrade` also provides a set of data analysis/visualization tools for the evaluation of historical performance as well as live environmental feedback. `FreqAI` builds on top of `Freqtrade` to include a user-friendly well tested interface for integrating external machine learning libraries for adaptive time-series forecasting. Beyond enabling the integration of existing libraries, `FreqAI` hosts a range of custom algorithms and methodologies aimed at improving computational and predictive performances. Thus, `FreqAI` contains a range of unique features which can be easily tested in combination with all the existing Python-accessible machine learning libraries to generate novel research on live and historical data.

The high-level overview of the software is depicted in Figure 1.

freqai-algo *Abstracted overview of FreqAI algorithm*

Connecting machine learning libraries

Although the `FreqAI` framework is designed to accommodate any Python library in the "Model training" and "Feature set engineering" portions of the software (Figure 1), it already boasts a wide range of well documented examples based on various combinations of:

- scikit-learn (Pedregosa et al., 2011), Catboost (Prokhorenkova et al., 2018), LightGBM (Ke et al., 2017), XGBoost (Chen & Guestrin, 2016), stable_baselines3 (Raffin et al., 2021), openai gym (Brockman et al., 2016), tensorflow (Abadi et al., 2015), pytorch (Paszke et al., 2019), Scipy (Virtanen et al., 2020), Numpy (Harris et al., 2020), and pandas (McKinney & others, 2010).

These mature projects contain a wide range of peer-reviewed and industry standard methods, including:

- Regression, Classification, Neural Networks, Reinforcement Learning, Support Vector Machines, Principal Component Analysis, point clustering, and much more.

which are all leveraged in `FreqAI` for users to use as templates or extend with their own methods.

Furnishing novel methods and features

Beyond the industry standard methods available through external libraries - `FreqAI` includes novel methods which are not available anywhere else in the open-source (or scientific) world. For example, `FreqAI` provides :

- a custom algorithm/methodology for adaptive modeling
- rapid and self-monitored feature engineering tools
- unique model features/indicators
- optimized data collection algorithms
- safely integrated outlier detection methods
- websocket communicated forecasts

Of particular interest for researchers, `FreqAI` provides the option of large scale experimentation via an optimized websocket communications interface.

Optimizing the back-end

`FreqAI` aims to make it simple for users to combine all the above tools to run studies based in two distinct modules:

- backtesting studies
- live-deployments

Both of these modules and their respective data management systems are built on top of `Freqtrade`, a mature and actively developed cryptocurrency trading software. This means that `FreqAI` benefits from a wide range of tangential/disparate feature developments such as:

- FreqUI, a graphical interface for backtesting and live monitoring
- telegram control
- robust database handling
- futures/leverage trading
- dollar cost averaging
- trading strategy handling
- a variety of free data sources via CCXT (FTX, Binance, Kucoin etc.)

These features derive from a strong external developer community that shares in the benefit and stability of a communal CI (Continuous Integration) system. Beyond the developer community, `FreqAI` benefits strongly from the userbase of `Freqtrade`, where most `FreqAI`

beta-testers/developers originated. This symbiotic relationship between `Freqtrade` and `FreqAI` ignited a thoroughly tested beta, which demanded a four month beta and comprehensive documentation containing:

- numerous example scripts
- a full parameter table
- methodological descriptions
- high-resolution diagrams/figures
- detailed parameter setting recommendations

Providing a reproducible foundation for researchers

`FreqAI` provides an extensible, robust, framework for researchers and citizen data scientists. The `FreqAI` sandbox enables rapid conception and testing of exotic hypotheses. From a research perspective, `FreqAI` handles the multitude of logistics associated with live deployments, historical backtesting, and feature engineering. With `FreqAI`, researchers can focus on their primary interests of feature engineering and hypothesis testing rather than figuring out how to collect and handle data. Further - the well maintained and easily installed open-source framework of `FreqAI` enables reproducible scientific studies. This reproducibility component is essential to general scientific advancement in time-series forecasting for chaotic systems.

Technical details

Typical users configure `FreqAI` via two files:

1. A `configuration` file (`--config`) which provides access to the full parameter list available here:

- control high-level feature engineering
- customize adaptive modeling techniques
- set any model training parameters available in third-party libraries
- manage adaptive modeling parameters (retrain frequency, training window size, continual learning, etc.)

2. A strategy file (`--strategy`) where users:

- list of the base training features
- set standard technical-analysis strategies
- control trade entry/exit criteria

With these two files, most users can exploit a wide range of pre-existing integrations in `Catboost` and 7 other libraries with a simple command:

```
freqtrade trade --config config_freqai.example.json --strategy FreqaiExampleStrategy --freqaimodel CatboostRegressor
```

Advanced users will edit one of the existing `--freqaimodel` files, which are simply an children of the `IFreqaiModel` (details below). Within these files, advanced users can customize training procedures, prediction procedures, outlier detection methods, data preparation, data saving methods, etc. This is all configured in a way where they can customize as little or as much as they want. This flexible customization is owed to the foundational architecture in `FreqAI`, which is comprised of three distinct Python objects:

- `IFreqaiModel`
  - A singular long-lived object containing all the necessary logic to collect data, store data, process data, engineer features, run training, and inference models.
- `FreqaiDataKitchen`
  - A short-lived object which is uniquely created for each asset/model. Beyond metadata, it also contains a variety of data processing tools.

- `FreqaiDataDrawer`
  - Singular long-lived object containing all the historical predictions, models, and save/load methods.

These objects interact with one another with one goal in mind - to provide a clean data set to machine learning experts/enthusiasts at the user endpoint. These power-users interact with an inherited `IFreqaiModel` that allows them to dig as deep or as shallow as they wish into the inheritence tree. Typical power-users focus their efforts on customizing training procedures and testing exotic functionalities available in third-party libraries. Thus, power-users are freed from the algorithmic weight associated with data management, and can instead focus their energy on testing creative hypotheses. Meanwhile, some users choose to override deeper functionalities within `IFreqaiModel` to help them craft unique data structures and training procedures.

The class structure and algorithmic details are depicted in the following diagram:

image *Class diagram summarizing object interactions in FreqAI*

Online documentation

The documentation for `FreqAI` is available online at https://www.freqtrade.io/en/latest/freqai/ and covers a wide range of materials:

- Quick-start with a single command and example files - (beginners)
- Introduction to the feature engineering interface and basic configurations - (intermediate users)
- Parameter table with indepth descriptions and default parameter setting recommendations - (intermediate users)
- Data analysis and post-processing - (advanced users)
- Methodological considerations complemented by high resolution figures - (advanced users)
- Instructions for integrating third party machine learning libraries into custom prediction models - (advanced users)
- Software architectural description with class diagram - (developers)
- File structure descriptions - (developers)

The docs direct users to a variety of pre-made examples which integrate `Catboost`, `LightGBM`, `XGBoost`, `Sklearn`, `stable_baselines3`, `torch`, `tensorflow`. Meanwhile, developers will also find thorough docstrings and type hinting throughout the source code to aid in code readability and customization.

`FreqAI` also benefits from a strong support network of users and developers on the `Freqtrade` discord as well as on the `FreqAI` discord. Within the `FreqAI` discord, users will find a deep and easily searched knowledge base containing common errors. But more importantly, users in the `FreqAI` discord share anectdotal and quantitative observations which compare performance between various third-party libraries and methods.

State of the field

There are two other open-source tools which are geared toward helping users build models for time-series forecasts on market based data. However, each of these tools suffer from a non-generalized frameworks that do not permit comparison of methods and libraries. Additionally, they do not permit easy live-deployments or adaptive-modeling methods. For example, two open-sourced projects called `tensortrade` (*Tensortrade*, 2022) and `FinRL` (*AI4Finance-Foundation*, 2022) limit users to the exploration of reinforcement learning on historical data. These softwares also do not provide robust live deployments, they do not furnish novel feature engineering algorithms, and they do not provide custom data analysis tools. `FreqAI` fills the gap.

On-going research

Emergent Methods, based in Arvada CO, is actively using `FreqAI` to perform large scale experiments aimed at comparing machine learning libraries in live and historical environments. Past projects include backtesting parametric sweeps, while active projects include a 3 week live deployment comparison between `CatboosRegressor`, `LightGBMRegressor`, and `XGBoostRegressor`. Results from these studies are on track for publication in scientific journals as well as more general data science blogs (e.g. Medium).

Installing and running `FreqAI`

`FreqAI` is automatically installed with `Freqtrade` using the following commands on linux systems:

```
git clone git@github.com:freqtrade/freqtrade.git
cd freqtrade
./setup.sh -i
```

However, `FreqAI` also benefits from `Freqtrade` docker distributions, and can be run with docker by pulling the stable or develop images from `Freqtrade` distributions.

Funding sources

`FreqAI` has had no official sponsors, and is entirely grass roots. All donations into the project (e.g. the GitHub sponsor system) are kept inside the project to help support development of open-sourced and communally beneficial features.

Acknowledgements

We would like to acknowledge various beta testers of `FreqAI`:

- Richárd Józsa
- Juha Nykänen
- Salah Lamkadem

As well as various `Freqtrade` developers maintaining tangential, yet essential, modules.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. https://www.tensorflow.org/

AI4Finance-foundation. (2022). https://github.com/AI4Finance-Foundation/FinRL

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI gym*. https://arxiv.org/abs/1606.01540

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant,

T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*, 3146–3154.

McKinney, W., & others. (2010). Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, *445*, 51–56.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased boosting with categorical features. *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 6639–6649.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, *22*(268), 1–8. http://jmlr.org/papers/v22/20-1364.html

Tensortrade. (2022). https://tensortradex.readthedocs.io/en/latest/L

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2