SeqMetrics: a unified library for performance metrics calculation in Python

Fazila Rubab [1], **Sara Iftikhar** [1], **and Ather Abbas** [2]

1 Environmental AI Research Group, Islamabad, Pakistan **2** King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

DOI: 10.21105/joss.06450

Software

- Review
- Repository
- Archive

Editor: Marcel Stimberg

Reviewers:

- @FATelarico
- @y1my1
- @SkafteNicki

Submitted: 16 February 2024
Published: 30 July 2024

Summary

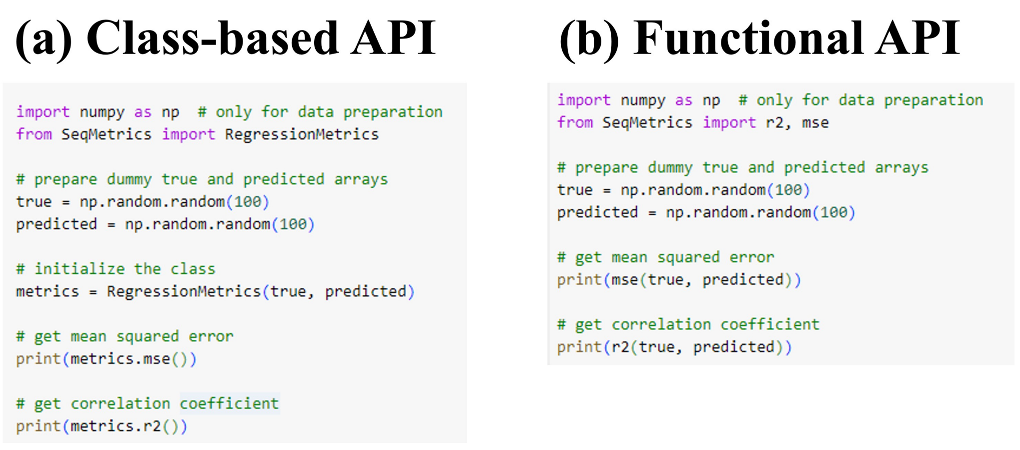
Current Python infrastructure lacks a robust, unified, and simplified library for error and performance metrics calculations. The SeqMetrics application responds to this critical need, providing a robust toolkit for evaluating regression and classification models with a comprehensive suite of performance metrics suitable for tabular and time series data. Designed for versatility, the library offers 112 regression and 22 classification metrics through both functional and class-based APIs. The design of the library ensures seamless integration into various coding environments. The web-based graphical user interface (GUI) of SeqMetrics enhances user accessibility, allowing efficient input through data arrays or file imports. It serves to users of varied programming expertise, offering a user-friendly interface for rigorous model assessment. The library prioritizes computational efficiency and has minimal dependencies with straightforward pip installation from PyPI. Rigorous testing of Seqmetrics ensures robustness, supported by extensive documentation for effective utilization by people from diverse backgrounds. Overall, SeqMetrics bridges the gap in Python's scientific analysis toolkit, contributing to data analysis.

Statement of need

Performance metrics and errors measure the distance and similarity between two arrays (Botchkarev, 2019). Quantification and analysis of model performance through error and performance metrics is a crucial step in any scientific modeling exercise (Gleckler et al., 2008). However, Python lacks a unified library encompassing performance metrics across diverse fields for one-dimensional numerical data. Existing Python libraries for performance evaluation offer only a limited number of metrics. For instance, the metrics sub-module from Keras (Chollet & others, 2015) contains only 24 metrics, while scikit-learn's (Pedregosa et al., 2011) metrics module covers 45 metrics. The TorchMetrics library (Detlefsen et al., 2022) offers a comprehensive list of over 100 metrics from diverse fields. However, it provides only 48 that are intended for 1-dimensional numerical data. Morevoer, the library's module-based API requires understanding of Object Oriented Programming and its simpler functional API does not cover all metrics. There are also some subject-specific libraries. These include NeuralHydrology (Kratzert et al., 2022), hydroeval (Hallouin, 2021), and HydroErr (Roberts et al., 2021), which address the needs of hydrology and water sciences. However, these resources lack more general sets of metrics, which the experts in these domains often require. Another limitation of these otherwise valuable libraries is their wide range of dependencies. For example, TorchMetrics depends on PyTorch (Paszke et al., 2019), which itself depends on several other libraries. This can lead to dependency conflicts due to version constraints, complicating the maintenance of multiple libraries within a single Python environment (Wang et al., 2020). Addressing this gap is imperative to provide researchers a unified platform for comprehensive model evaluation and with minimal requirements, streamlining their computational workflows and enhancing accuracy and reproducibility across various domains (Mukherjee et al., 2021). The SeqMetrics application addresses the critical need for a robust and versatile toolkit for assessing and comparing regression and classification model performance across a spectrum of domains. With a comprehensive suite of performance metrics for sequential (tabular and time series) data, spanning traditional statistical measures to specialized atmospheric sciences metrics, the software serves as a valuable resource for researchers, analysts, and practitioners in fields such as hydrology, finance, and engineering. By providing a standardized platform for evaluating model performance through a diverse set of metrics, the application facilitates the rigorous validation and optimization of regression and classification models, contributing to informed decision-making processes and ensuring the reliability of predictive modeling in complex and dynamic systems.

API design

The SeqMetrics library offers a comprehensive suite of 112 regression metrics and 22 classification metrics to facilitate a detailed assessment of model performance. Regression metrics such as R-squared, Mean Squared Error (MSE), and Root MSE are provided for use with continuous variables. On the other hand, for categorical data, there are classification metrics including accuracy, precision, recall, F1 score, and the area under the ROC curve, among others. From a programming perspective, the SeqMetrics library employs a modular architecture, offering functional and class-based APIs for smooth integration across diverse coding environments. The class-based API consists of `RegressionMetrics` and `ClassificationMetrics` classes (Figure 1a), which provide users with a structured approach for model evaluation. The user has to first initialize these classes by providing the data and then all the performance metrics are available from the respective instances of the classes. Conversely, the functional API offers a more simplified approach to access these metrics without initializing the classes initially (Figure 1b). All the functions which calculate performance metrics receive two arrays as obligatory input arguments and return a scalar value as output. With a unified API design and minimal dependency only on NumPy, the library prioritizes efficiency in computational tasks. It ensures straightforward installation via pip from the Python Package Index (PyPI), a widely adopted standard, which not only streamlines the process but also contributes to the overall efficiency of the application for the broader scientific community.


(a) Class-based API

```python
import numpy as np  # only for data preparation
from SeqMetrics import RegressionMetrics

# prepare dummy true and predicted arrays
true = np.random.random(100)
predicted = np.random.random(100)

# initialize the class
metrics = RegressionMetrics(true, predicted)

# get mean squared error
print(metrics.mse())

# get correlation coefficient
print(metrics.r2())
```

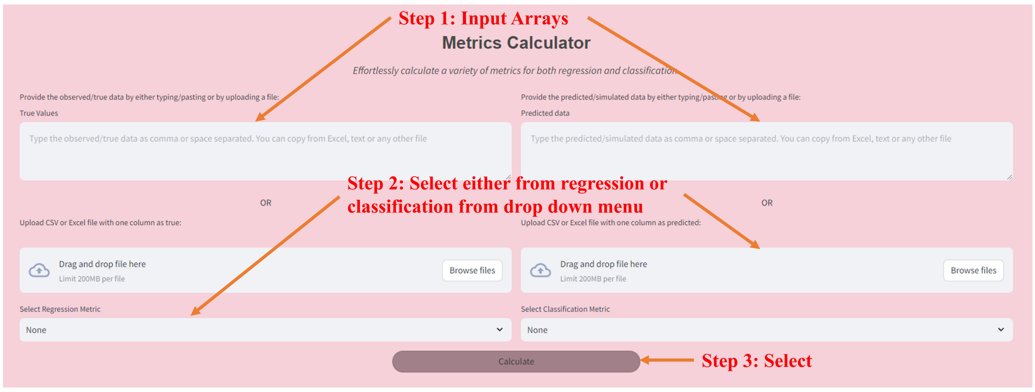
(b) Functional API

```python
import numpy as np  # only for data preparation
from SeqMetrics import r2, mse

# prepare dummy true and predicted arrays
true = np.random.random(100)
predicted = np.random.random(100)

# get mean squared error
print(mse(true, predicted))

# get correlation coefficient
print(r2(true, predicted))
```

Figure 1: Comparison of class-based and functional API

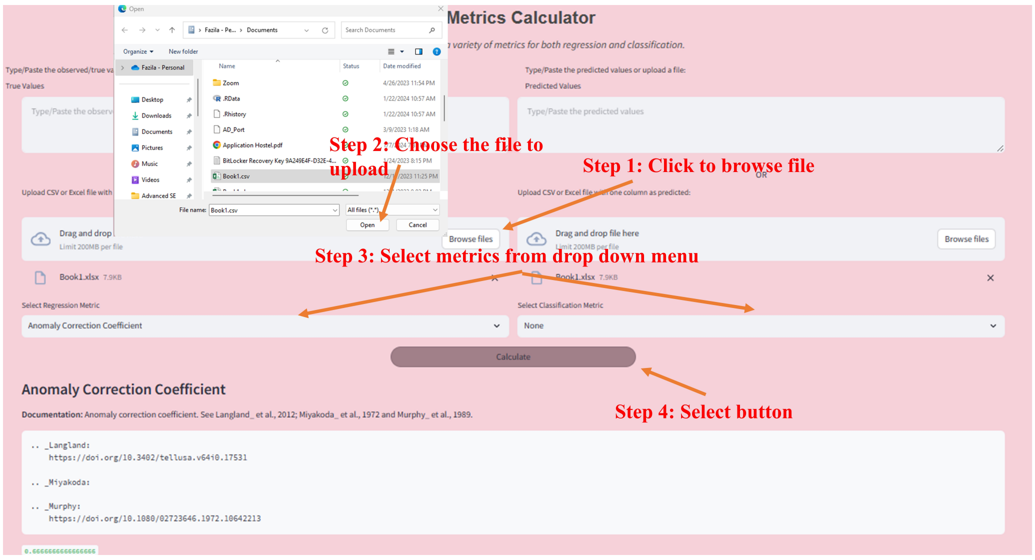
Graphical User Interface

The SeqMetrics GUI offers a user-friendly and intuitive platform for seamless error calculation. This interface is built and deployed using streamlit at https://seqmetrics.streamlit.app. There are two ways of providing input in this web-based GUI. The first method consists of providing the input data arrays by copying and pasting the true and predicted arrays (Figure 2). Another way is by importing CSV or Excel files into the interface (Figure 3). This streamlines the process of entering true and predicted values for evaluation. The GUI provides a clean and organized layout, guiding users through the evaluation workflow with clear instructions. With its simplicity and ease of use, the GUI empowers users to perform regression and classification model assessments effortlessly. The SeqMetrics GUI enhances accessibility and efficiency in evaluating model performance for both seasoned data scientists and beginners, without compromising on robustness and precision. Therefore, the design of the SeqMetrics is equally beneficial for advanced programmers as well as for those with limited programming knowledge.



Step 1: Input Arrays
Metrics Calculator
Effortlessly calculate a variety of metrics for both regression and classification
Provide the observed/true data by either typing/pasting or by uploading a file:
True Values
Type the observed/true data as comma or space separated. You can copy from Excel, text or any other file
Provide the predicted/simulated data by either typing/pasting or by uploading a file:
Predicted data
Type the predicted/simulated data as comma or space separated. You can copy from Excel, text or any other file
Step 2: Select either from regression or classification from drop down menu
OR
Upload CSV or Excel file with one column as true:
OR
Upload CSV or Excel file with one column as predicted:
Drag and drop file here
Limit 200MB per file
Browse files
Drag and drop file here
Limit 200MB per file
Browse files
Select Regression Metric
None
Select Classification Metric
None
Calculate
Step 3: Select

Figure 2: Method of copying and pasting arrays in SeqMetrics GUI



Open
Metrics Calculator
variety of metrics for both regression and classification.
Type/Paste the observed/true va
True Values
Type/Paste the observ
Type/Paste the predicted values or upload a file:
Predicted Values
Type/Paste the predicted values
Step 2: Choose the file to upload
Step 1: Click to browse file
Upload CSV or Excel file with
Upload CSV or Excel file with one column as predicted:
File name: Book1.csv
All files (*.*)
Open
Cancel
Drag and drop
Limit 200MB per file
Browse files
Drag and drop file here
Limit 200MB per file
Browse files
Step 3: Select metrics from drop down menu
Book1.xlsx 7.9KB
Book1.xlsx 7.9KB
Select Regression Metric
Anomaly Correction Coefficient
Select Classification Metric
None
Calculate
Step 4: Select button

Anomaly Correction Coefficient

Documentation: Anomaly correction coefficient. See Langland_ et al., 2012; Miyakoda_ et al., 1972 and Murphy_ et al., 1989.

```
.. _Langland:
    https://doi.org/10.3402/tellusa.v64i0.17531

.. _Miyakoda:

.. _Murphy:
    https://doi.org/10.1080/02723646.1972.10642213
```

```
0.6666666666666666
```

Figure 3: Method of reading data from files in SeqMetrics GUI

The streamlit-based GUI can also be launched locally without having to upload the data on streamlit servers. This can be useful for users with data-privacy concerns or those without internet connection. The steps to launch streamlit GUI locally involve cloning the respository, installing the requirements and streamlit Python package and then launching the streamlit app. These steps are given below:

```
git clone https://github.com/AtrCheema/SeqMetrics.git
cd SeqMetrics
```

```
pip install -r requirements.txt
pip install streamlit
streamlit run app.py
```

Testing and documentation

Following the 'unit test' protocol, the library undergoes comprehensive testing of all regression and classification metrics. The library is tested for multiple scenarios especially for the classification case which includes numerical and logits inputs, ensuring robustness in various classification contexts such as binary, multiclass, and multilabel. Such comprehensive testing ensures the robustness and accuracy of each metric, providing users with reliable results. Additionally, the library features extensive documentation, with detailed docstrings for each function and class, accompanied by fully executable examples. This thorough documentation enhances user understanding and facilitates efficient utilization of the library's capabilities in both regression and classification scenarios.

References

Botchkarev, A. (2019). Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *Interdisciplinary Journal of Information, Knowledge, and Management*, *14*, 45–76. https://doi.org/10.28945/4184

Chollet, F., & others. (2015). *Keras*. https://github.com/fchollet/keras; GitHub.

Detlefsen, N. S., Borovec, J., Schock, J., Jha, A. H., Koker, T., Di Liello, L., Stancl, D., Quan, C., Grechkin, M., & Falcon, W. (2022). TorchMetrics - measuring reproducibility in PyTorch. *Journal of Open Source Software*, *7*(70), 4101. https://doi.org/10.21105/joss.04101

Gleckler, P. J., Taylor, K. E., & Doutriaux, C. (2008). Performance metrics for climate models. *Journal of Geophysical Research: Atmospheres*, *113*(D6). https://doi.org/10.1029/2007JD008972

Hallouin, T. (2021). *Hydroeval: An evaluator for streamflow time series in Python*. GitHub. https://doi.org/10.5281/zenodo.2591217

Kratzert, F., Gauch, M., Nearing, G., & Klotz, D. (2022). NeuralHydrology — a Python library for deep learning research in hydrology. *Journal of Open Source Software*, *7*(71), 4050. https://doi.org/10.21105/joss.04050

Mukherjee, S., Almanza, A., & Rubio-González, C. (2021). Fixing dependency errors for Python build reproducibility. *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 439–451. https://doi.org/10.1145/3460319.3464797

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., & others. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*. https://doi.org/10.48550/arXiv.1912.01703

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

Roberts, W., Williams, G., Jackson, E., Nelson, J., & Ames, D. (2021). Hydrostats: A Python package for characterizing errors between observed and predicted time series. *Hydrology*, *5*. https://doi.org/10.3390/hydrology5040066

Wang, Y., Wen, M., Liu, Y., Wang, Y., Li, Z., Wang, C., Yu, H., Cheung, S.-C., Xu, C., &

Zhu, Z. (2020). Watchman: Monitoring dependency conflicts for Python library ecosystem. *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 125–135. https://doi.org/10.1145/3377811.3380426