# permetrics: A Framework of Performance Metrics for Machine Learning Models

Nguyen Van Thieu ⓘ [1]

1 Faculty of Computer Science, Phenikaa University, Yen Nghia, Ha Dong, Hanoi, 12116, Vietnam.

Summary

Performance metrics are pivotal in machine learning field, especially for tasks like regression, classification, and clustering (Saura, 2021). They offer quantitative measures to assess the accuracy and efficacy of models, aiding researchers and practitioners in evaluating, contrasting, and enhancing algorithms and models. In regression tasks, where continuous predictions are made, metrics such as mean squared error (MSE), root mean square error (RMSE), and Coefficient of Determination (COD) (Nguyen et al., 2018; Nguyen, Nguyen, & Nguyen, 2019) can reveal how well models capture data patterns. In classification tasks, metrics such as accuracy, precision, recall, F1-score, and AUC-ROC (Luque et al., 2019) assess a model's ability to classify instances correctly, detect false results, and gauge overall predictive performance. Clustering tasks aim to discover inherent patterns and structures within unlabeled data by grouping similar instances together. Metrics like Silhouette coefficient, Davies-Bouldin index, and Calinski-Harabasz index (Nainggolan et al., 2019) measure clustering quality, helping evaluate how well algorithms capture data distribution and assign instances to clusters. In general, performance metrics serve multiple purposes. They enable researchers to compare different models and algorithms (Ahmed et al., 2021), identify strengths and weaknesses (Nguyen, Nguyen, Nguyen, & Nguyen, 2019), and make informed decisions about model selection and parameter tuning (Nguyen, Hoang, et al., 2020). Moreover, it also plays a crucial role in the iterative process of model development and improvement. By quantifying the model's performance, metrics guide the optimization process (Van Thieu, Deb Barma, et al., 2023), allowing researchers to fine-tune algorithms, explore feature engineering techniques (Nguyen et al., 2021), and address issues such as overfitting, underfitting, and bias (Nguyen, Nguyen, et al., 2020). This paper introduces a Python framework named **PerMetrics** (PERformance METRICS), designed to offer comprehensive performance metrics for machine learning models. The library, packaged as `permetrics`, is open-source and written in Python. It provides a wide number of metrics to enable users to evaluate their models effectively. `permetrics` is hosted on GitHub and is under continuous development and maintenance by the dedicated team. The framework is accompanied by comprehensive documentation, examples, and test cases, facilitating easy comprehension and integration into users' workflows.

Statement of need

PerMetrics is a Python project developed in the field of performance assessment and machine learning. To the best of our knowledge, it is the first open-source framework that contributes a significant number of metrics, totaling 111 methods, for three fundamental problems: regression, classification, and clustering. This library relies exclusively on only two well-known third-party Python scientific computing packages: `NumPy` (Harris et al., 2020) and `SciPy` (Virtanen et al., 2020). The modules of `permetrics` are extensively documented, and the automatically generated API provides a complete and up-to-date description of both the object-oriented and functional implementations underlying the framework.

⁴³ To gain a better understanding of the necessity of `permetrics` library, this section will compare
⁴⁴ it to several notable libraries currently are available. Most notably, `Scikit-Learn` (Pedregosa et
⁴⁵ al., 2011), which also encompasses an assortment of metrics for regression, classification, and
⁴⁶ clustering problems. Nevertheless, a few classification metrics present in `Scikit-Learn` lack
⁴⁷ support for multiple outputs, such as the Matthews correlation coefficient (MCC) and Hinge
⁴⁸ loss. Furthermore, critical metrics such as RMSE, mean absolute percentage error (MAPE),
⁴⁹ Nash-Sutcliffe efficiency (NSE), and Kling-Gupta efficiency (KGE) are absent. `permetrics`
⁵⁰ addresses these deficiencies. Additionally, `Scikit-Learn` is deficient in various vital clustering
⁵¹ metrics, including but not limited to Ball Hall index, Banfeld Raftery index, sum of squared
⁵² error, Duda Hart index, and Hartigan index (Van Thieu, Oliva, et al., 2023).

⁵³ Another popular package is `Metrics` (Hamner, 2015). It provides a variety of metrics for
⁵⁴ different programming languages such as Python, MATLAB, R, and Haskell. However, the
⁵⁵ development team has ceased activity since 2015. They offer a limited number of metrics
⁵⁶ because they focused on creating a single set of metrics for multiple programming languages.
⁵⁷ Additionally, the metrics are not packaged as a complete library but rather exist as repository
⁵⁸ code on GitHub.

⁵⁹ `TorchMetrics` (Nicki Skafte Detlefsen et al., 2022) is a widely recognized framework for
⁶⁰ performance metrics developed for PyTorch users. The library includes over 100 metrics,
⁶¹ covering various domains such as regression, classification, audio, detection, and text. However,
⁶² `TorchMetrics` does not provide metrics specifically for clustering tasks. Although it offers
⁶³ a substantial number of metrics, it falls short compared to `permetrics`. Moreover, it relies
⁶⁴ heavily on other major libraries such as `NumPy`, `Torch`, `Typing-extensions`, `Packaging`, and
⁶⁵ `Lightning-utilities`. Additionally, using this library may not be easy for beginners in Python
⁶⁶ programming, as it requires a deep understanding of the Torch library to utilize `TorchMetrics`
⁶⁷ effectively.

⁶⁸ Other popular libraries such as `TensorFlow` (Abadi et al., 2016), `Keras` (Chollet, 2017),
⁶⁹ `CatBoost` (Prokhorenkova et al., 2018), and `MxNet` (Chen et al., 2015) also contain modules
⁷⁰ dedicated to metrics. However, the issue with these libraries is that their metric modules are
⁷¹ specific to each respective one. It is challenging to combine metric modules from different
⁷² libraries with each other. If it is possible to combine them, it often requires installing numerous
⁷³ related libraries. Furthermore, the metric modules within each library are tailored to users who
⁷⁴ are familiar with that specific one, requiring users to learn multiple libraries, syntax structures,
⁷⁵ and necessary commands associated with each framework to use them in combination. These
⁷⁶ are significant obstacles when using metrics from such libraries.

⁷⁷ All the aforementioned challenges are addressed by our `permetrics` library. It not only offers
⁷⁸ a simple and concise syntax and usage but also does not require any knowledge of other
⁷⁹ major libraries such as `TensorFlow`, `Keras`, or `PyTorch`. Additionally, it can be seamlessly
⁸⁰ integrated with any computational or machine learning library. In the future, we plan to expand
⁸¹ `permetrics` to include other domains such as text metrics, audio metrics, detection metrics,
⁸² and image metrics.

⁸³ Available Methods

⁸⁴ At the time of publication, `PerMetrics` provides three types of performance metrics include
⁸⁵ regression, classification, and clustering metrics. We listed all methods of each type below.

Problem	ID	Metric	Metric Fullname
****	1	EVS	Explained Variance Score
****	2	ME	Max Error
****	3	MBE	Mean Bias Error
****	4	MAE	Mean Absolute Error

Problem	ID	Metric	Metric Fullname
****	5	MSE	Mean Squared Error
****	6	RMSE	Root Mean Squared Error
****	7	MSLE	Mean Squared Log Error
****	8	MedAE	Median Absolute Error
****	9	MRE / MRB	Mean Relative Error / Mean Relative Bias
****	10	MPE	Mean Percentage Error
****	11	MAPE	Mean Absolute Percentage Error
****	12	SMAPE	Symmetric Mean Absolute Percentage Error
****	13	MAAPE	Mean Arctangent Absolute Percentage Error
****	14	MASE	Mean Absolute Scaled Error
****	15	NSE	Nash-Sutcliffe Efficiency Coefficient
****	16	NNSE	Normalized Nash-Sutcliffe Efficiency Coefficient
****	17	WI	Willmott Index
****	18	R / PCC	Pearson's Correlation Coefficient
****	19	AR / APCC	Absolute Pearson's Correlation Coefficient
****	20	RSQ/R2S	(Pearson's Correlation Index) $\hat{}$ 2
****	21	R2 / COD	Coefficient of Determination
****	22	AR2 / ACOD	Adjusted Coefficient of Determination
****	23	CI	Confidence Index
****	24	DRV	Deviation of Runoff Volume
****	25	KGE	Kling-Gupta Efficiency
****	26	GINI	Gini Coefficient
****	27	GINI_WIKI	Gini Coefficient on Wikipage
****	28	PCD	Prediction of Change in Direction
****	29	CE	Cross Entropy
****	30	KLD	Kullback Leibler Divergence
****	31	JSD	Jensen Shannon Divergence
****	32	VAF	Variance Accounted For
****	33	RAE	Relative Absolute Error
****	34	A10	A10 Index
****	35	A20	A20 Index
****	36	A30	A30 Index
****	37	NRMSE	Normalized Root Mean Square Error
****	38	RSE	Residual Standard Error
****	39	RE / RB	Relative Error / Relative Bias
****	40	AE	Absolute Error
****	41	SE	Squared Error
****	42	SLE	Squared Log Error
****	43	COV	Covariance
****	44	COR	Correlation
****	45	EC	Efficiency Coefficient
****	46	OI	Overall Index
****	47	CRM	Coefficient of Residual Mass
–	–	–	–
Classification	1	PS	Precision Score
****	2	NPV	Negative Predictive Value
****	3	RS	Recall Score
****	4	AS	Accuracy Score
****	5	F1S	F1 Score
****	6	F2S	F2 Score
****	7	FBS	F-Beta Score

Problem	ID	Metric	Metric Fullname
****	8	SS	Specificity Score
****	9	MCC	Matthews Correlation Coefficient
****	10	HS	Hamming Score
****	11	CKS	Cohen's kappa score
****	12	JSI	Jaccard Similarity Coefficient
****	13	GMS	Geometric Mean Score
****	14	ROC-AUC	ROC-AUC
****	15	LS	Lift Score
****	16	GINI	GINI Index
****	17	CEL	Cross Entropy Loss
****	18	HL	Hinge Loss
****	19	KLDL	Kullback Leibler Divergence Loss
****	20	BSL	Brier Score Loss
–	–	–	–
Clustering	1	BHI	Ball Hall Index
****	2	XBI	Xie Beni Index
****	3	DBI	Davies Bouldin Index
****	4	BRI	Banfeld Raftery Index
****	5	KDI	Ksq Detw Index
****	6	DRI	Det Ratio Index
****	7	DI	Dunn Index
****	8	CHI	Calinski Harabasz Index
****	9	LDRI	Log Det Ratio Index
****	10	LSRI	Log SS Ratio Index
****	11	SI	Silhouette Index
****	12	SSEI	Sum of Squared Error Index
****	13	MSEI	Mean Squared Error Index
****	14	DHI	Duda-Hart Index
****	15	BI	Beale Index
****	16	RSI	R-squared Index
****	17	DBCVI	Density-based Clustering Validation Index
****	18	HI	Hartigan Index
****	19	MIS	Mutual Info Score
****	20	NMIS	Normalized Mutual Info Score
****	21	RaS	Rand Score
****	22	ARS	Adjusted Rand Score
****	23	FMS	Fowlkes Mallows Score
****	24	HS	Homogeneity Score
****	25	CS	Completeness Score
****	26	VMS	V-Measure Score
****	27	PrS	Precision Score
****	28	ReS	Recall Score
****	29	FmS	F-Measure Score
****	30	CDS	Czekanowski Dice Score
****	31	HGS	Hubert Gamma Score
****	32	JS	Jaccard Score
****	33	KS	Kulczynski Score
****	34	MNS	Mc Nemar Score
****	35	PhS	Phi Score
****	36	RTS	Rogers Tanimoto Score
****	37	RRS	Russel Rao Score
****	38	SS1S	Sokal Sneath1 Score
****	39	SS2S	Sokal Sneath2 Score

Problem	ID	Metric	Metric Fullname
****	40	PuS	Purity Score
****	41	ES	Entropy Score
****	42	TS	Tau Score
****	43	GAS	Gamma Score
****	44	GPS	Gplus Score

Installation and Simple Example

PerMetrics is published to the Python Packaging Index (PyPI) and can be installed via pip

```
pip install permetrics
```

Below are a few fundamental examples illustrating the usage of the PerMetrics library. We have prepared a folder examples in Github repository that contains these examples and more advances one. Furthermore, to gain a comprehensive understanding of our library, we recommend reading the documentation available at the following link.

Regression Metrics

```python
import numpy as np
from permetrics.regression import RegressionMetric

y_true = np.array([3, -0.5, 2, 7, 5, 6])
y_pred = np.array([2.5, 0.0, 2, 8, 5, 6])

evaluator = RegressionMetric(y_true=y_true, y_pred=y_pred)

print(evaluator.mean_squared_error())
print(evaluator.median_absolute_error())
print(evaluator.MAPE())
```

Classification Metrics

```python
from permetrics.classification import ClassificationMetric

## For integer labels or categorical labels
y_true = [0, 1, 0, 0, 1, 0]
y_pred = [0, 1, 0, 0, 0, 1]

# y_true = ["cat", "ant", "cat", "cat", "ant", "bird", "bird", "bird"]
# y_pred = ["ant", "ant", "cat", "cat", "ant", "cat", "bird", "ant"]

evaluator = ClassificationMetric(y_true, y_pred)

print(evaluator.f1_score())
print(evaluator.F1S(average="micro"))
print(evaluator.f1_score(average="macro"))
```

Clustering Metrics

```python
import numpy as np
from permetrics import ClusteringMetric
from sklearn.datasets import make_blobs
```

```
# generate sample data
X, y_true = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)
y_pred = np.random.randint(0, 4, size=300)

evaluator = ClusteringMetric(y_true=y_true, y_pred=y_pred, X=X)

# Call specific function inside object, each function has 2 names (fullname and short na

##      + Internal metrics: Need X and y_pred and has suffix as index
print(evaluator.ball_hall_index(X=X, y_pred=y_pred))
print(evaluator.CHI(X=X, y_pred=y_pred))

##      + External metrics: Need y_true and y_pred and has suffix as score
print(evaluator.adjusted_rand_score(y_true=y_true, y_pred=y_pred))
print(evaluator.completeness_score(y_true=y_true, y_pred=y_pred))
```

Acknowledgements

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., & others. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv Preprint arXiv:1603.04467*. https://doi.org/10.48550/arXiv.1603.04467

Ahmed, A. N., Van Lam, T., Hung, N. D., Van Thieu, N., Kisi, O., & El-Shafie, A. (2021). A comprehensive comparison of recent developed meta-heuristic algorithms for streamflow time series forecasting problem. *Applied Soft Computing*, *105*, 107282. https://doi.org/10.1016/j.asoc.2021.107282

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., & Zhang, Z. (2015). Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv Preprint arXiv:1512.01274*. https://doi.org/10.48550/arXiv.1512.01274

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1251–1258. https://doi.org/10.1109/CVPR.2017.195

Hamner, B. (2015). *Metrics*. https://github.com/benhamner/Metrics

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., & others. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Luque, A., Carrasco, A., Martín, A., & De Las Heras, A. (2019). The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, *91*, 216–231. https://doi.org/10.1016/j.patcog.2019.02.023

Nainggolan, R., Perangin-angin, R., Simarmata, E., & Tarigan, A. F. (2019). Improved the Performance of the K-Means Cluster Using the Sum of Squared Error (SSE) optimized by using the Elbow Method. *Journal of Physics: Conference Series*, *1361*(1), 012015. https://doi.org/10.1088/1742-6596/1361/1/012015

Nguyen, T., Hoang, B., Nguyen, G., & Nguyen, B. M. (2020). A new workload prediction model using extreme learning machine and enhanced tug of war optimization. *Procedia Computer Science*, *170*, 362–369. https://doi.org/10.1016/j.procs.2020.03.063

Nguyen, T., Nguyen, B. M., & Nguyen, G. (2019). Building resource auto-scaler with functional-link neural network and adaptive bacterial foraging optimization. *International Conference on Theory and Applications of Models of Computation*, 501–517. https://doi.org/10.1007/978-3-030-14812-6_31

Nguyen, T., Nguyen, G., & Nguyen, B. M. (2020). EO-CNN: An enhanced CNN model trained by equilibrium optimization for traffic transportation prediction. *Procedia Computer Science*, *176*, 800–809. https://doi.org/10.1016/j.procs.2020.09.075

Nguyen, T., Nguyen, T., Nguyen, B. M., & Nguyen, G. (2019). Efficient time-series forecasting using neural network and opposition-based coral reefs optimization. *International Journal of Computational Intelligence Systems*, *12*, 1144–1161. https://doi.org/10.2991/ijcis.d.190930.003

Nguyen, T., Nguyen, T., Vu, Q.-H., Huynh, T. T. B., & Nguyen, B. M. (2021). Multi-objective sparrow search optimization for task scheduling in fog-cloud-blockchain systems. *2021 IEEE International Conference on Services Computing (SCC)*, 450–455. https://doi.org/10.1109/scc53864.2021.00065

Nguyen, T., Tran, N., Nguyen, B. M., & Nguyen, G. (2018). A resource usage prediction system using functional-link and genetic algorithm neural network for multivariate cloud metrics. *2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA)*, 49–56. https://doi.org/10.1109/SOCA.2018.00014

Nicki Skafte Detlefsen, Jiri Borovec, Justus Schock, Ananya Harsh, Teddy Koker, Luca Di Liello, Daniel Stancl, Changsheng Quan, Maxim Grechkin, & William Falcon. (2022). *TorchMetrics - Measuring Reproducibility in PyTorch*. https://doi.org/10.21105/joss.04101

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. https://doi.org/10.48550/arXiv.1201.0490

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, *31*. https://doi.org/10.48550/arXiv.1706.09516

Saura, J. R. (2021). Using Data Sciences in Digital Marketing: Framework, methods, and performance metrics. *Journal of Innovation & Knowledge*, *6*(2), 92–102. https://doi.org/10.1016/j.jik.2020.08.001

Van Thieu, N., Deb Barma, S., Van Lam, T., Kisi, O., & Mahesha, A. (2023). Groundwater level modeling using Augmented Artificial Ecosystem Optimization. *Journal of Hydrology*, *617*, 129034. https://doi.org/10.1016/j.jhydrol.2022.129034

Van Thieu, N., Oliva, D., & Pérez-Cisneros, M. (2023). MetaCluster: An open-source python library for metaheuristic-based clustering problems. *SoftwareX*, *24*, 101597. https://doi.org/10.1016/j.softx.2023.101597

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., & others. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2