The Journal of Open Source Software

MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack

Sebastian Raschka[1]

1 Michigan State University

DOI: 10.21105/joss.00638

Software

- Review
- Repository
- Archive

Submitted: 15 March 2018
Published: 21 March 2018

Summary

MLxtend is a library that implements a variety of core algorithms and utilities for machine learning and data mining. The primary goal of MLxtend is to make commonly used tools accessible to researchers in academia and data scientists in industries focussing on user-friendly and intuitive APIs and compatibility to existing machine learning libraries, such as scikit-learn, when appropriate. While MLxtend implements a large variety of functions, highlights include sequential feature selection algorithms (Pudil, Novovičová, and Kittler 1994), implementations of stacked generalization (Wolpert 1992) for classification and regression, and algorithms for frequent pattern mining (Agrawal, Srikant, and others 1994). The sequential feature selection algorithms cover forward, backward, forward floating, and backward floating selection and leverage scikit-learn's cross-validation API (Pedregosa et al. 2011) to ensure satisfactory generalization performance upon constructing and selecting feature subsets. Besides, visualization functions are provided that allow users to inspect the estimated predictive performance, including performance intervals, for different feature subsets. The ensemble methods in MLxtend cover majority voting, stacking, and stacked generalization, all of which are compatible with scikit-learn estimators and other libraries as XGBoost (Chen and Guestrin 2016). In addition to feature selection, classification, and regression algorithms, MLxtend implements model evaluation techniques for comparing the performance of two different models via McNemar's test and multiple models via Cochran's Q test. An implementation of the 5x2 cross-validated paired t-test (Dietterich 1998) allows users to compare the performance of machine learning algorithms to each other. Furthermore, different flavors of the Bootstrap method (Efron and Tibshirani 1994), such as the .632 Bootstrap method (Efron 1983) are implemented to compute confidence intervals of performance estimates. All in all, MLxtend provides a large variety of different utilities that build upon and extend the capabilities of Python's scientific computing stack.

References

Agrawal, Rakesh, Ramakrishnan Srikant, and others. 1994. "Fast Algorithms for Mining Association Rules." In *Proc. 20th Int. Conf. Very Large Data Bases, Vldb*, 1215:487–99.

Chen, Tianqi, and Carlos Guestrin. 2016. "Xgboost: A Scalable Tree Boosting System." In *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 785–94. ACM.

Dietterich, Thomas G. 1998. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms." *Neural Computation* 10 (7). MIT Press:1895–1923.

Efron, Bradley. 1983. "Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation." *Journal of the American Statistical Association* 78 (382). Taylor & Francis Group:316–31.

Efron, Bradley, and Robert J Tibshirani. 1994. *An Introduction to the Bootstrap*. CRC press.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12 (Oct):2825–30.

Pudil, Pavel, Jana Novovičová, and Josef Kittler. 1994. "Floating Search Methods in Feature Selection." *Pattern Recognition Letters* 15 (11). Elsevier:1119–25.

Wolpert, David H. 1992. "Stacked Generalization." *Neural Networks* 5 (2). Elsevier:241–59.