The Journal of Open Source Software

Yellowbrick: Visualizing the Scikit-Learn Model Selection Process

Benjamin Bengfort[1] **and Rebecca Bilbro**[1]

1 Georgetown University

DOI: 10.21105/joss.01075

Software

- Review
- Repository
- Archive

Submitted: 30 July 2018
Published: 24 March 2019

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC-BY).

Summary

Discussions of machine learning are frequently characterized by a singular focus on algorithmic behavior. Be it logistic regression, random forests, Bayesian methods, or artificial neural networks, practitioners are often quick to express their preference. However, model selection is more nuanced than simply picking the "right" or "wrong" algorithm. In practice, the workflow includes multiple iterations through feature engineering, algorithm selection, and hyperparameter tuning — summarized by Kumar et al. as a search for the maximally performing model selection triple (Kumar, McCann, Naughton, & Patel, 2016). "Model selection," they explain, "is iterative and exploratory because the space of [model selection triples] is usually infinite, and it is generally impossible for analysts to know a priori which [combination] will yield satisfactory accuracy and/or insights."

Treating model selection as search has led to automation through grid search methods, standardized APIs, drag and drop GUIs, and specialized database systems. However, the search problem is computationally intractable and research in both machine learning (Wickham, Cook, & Hofmann, 2015) and visual analytics (Liu, Wang, Liu, & Zhu, 2017) suggests human intuition and guidance can more effectively hone in on quality models than exhaustive optimization methods. By visualizing the model selection process, data scientists can interactively steer towards final, interpretable models and avoid pitfalls and traps (Kapoor, Lee, Tan, & Horvitz, 2010).

Yellowbrick is a response to the call for open source visual steering tools. For data scientists, Yellowbrick helps evaluate the stability and predictive value of machine learning models and improves the speed of the experimental workflow. For data engineers, Yellowbrick provides visual tools for monitoring model performance in real world applications. For users of models, Yellowbrick provides visual interpretation of the behavior of the model in high dimensional feature space. Finally, for students, Yellowbrick is a framework for understanding a large variety of algorithms and methods.

Implemented in Python, the Yellowbrick visualization package achieves steering by extending both scikit-learn (Pedregosa et al., 2011) and Matplotlib (Hunter, 2007). Like Yellowbrick, both scikit-learn and Matplotlib are extensions of SciPy (Jones, Oliphant, Peterson, & others, n.d.), libraries intended to facilitate scientific computing. Scikit-learn provides a generalized API for machine learning by exposing the concept of an `Estimator`, an object that learns from data. Yellowbrick in turn extends this concept with the idea of a `Visualizer`, an object that both learns from data and visualizes the result. Visualizers wrap Matplotlib procedures to produce publication-ready figures and rich visual analytics.

Because Yellowbrick is part of a rich visual and machine learning ecosystem, it provides visualizations for feature and target analysis, classification, regression, and clustering model visualization, hyperparameter tuning, and text analysis. A few selected examples of visual diagnostics for model selection and their interpretations follow.

The Journal of Open Source Software

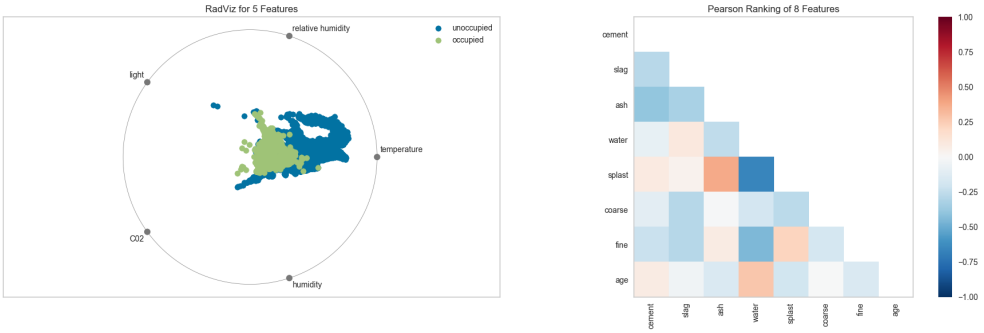

Figure 1: Feature Analysis

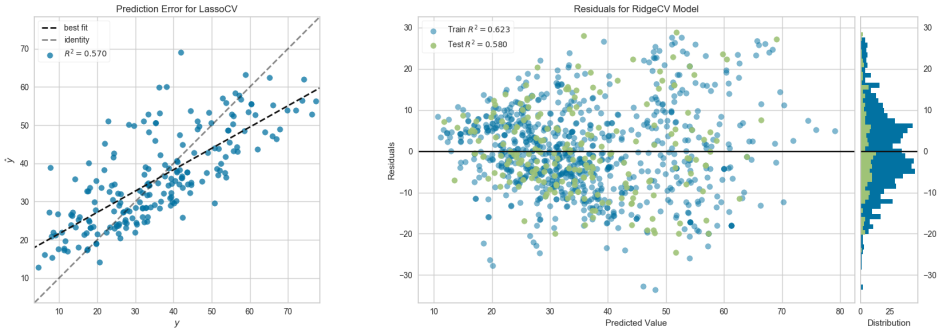

Figure 2: Regression Model Tuning

Because "more data beats better algorithms" (Rajaraman, 2008), the first step to creating valid, predictive models is to find the minimum set of features that predicts the dependent variable. Generally, this means finding features that describe data in high dimensional space that are *separable* (i.e., by a hyperplane). Tools like `RadViz`, `ParallelCoordinates`, and `Manifold` help visualize high dimensional data for quick diagnostics. Bayesian models and regressions suffer when independent variables are collinear (i.e., exhibit pairwise correlation). `Rank2D` visualizations show pairwise correlations among features and can facilitate feature elimination.

Regression models hypothesize some underlying function influenced by noise whose central tendency can be inferred. The `PredictionError` visualizer shows the relationship of actual to predicted values, giving a sense of heteroskedasticity in the target, or regions of more or less error as predictions deviate from the 45 degree line. The `ResidualsPlot` shows the relationship of error in the training and test data and can also show regions of increased variability in the predictive model.

Classification analysis focuses on the precision and recall of the model's prediction of individual classes. The `ClassificationReport` visualizer allows for rapid comparison between models as a visual heatmap of these metrics. The `DiscriminationThreshold` visualizer for binary classifiers shows how adjusting the threshold for positive classification may influence precision and recall globally, as well as the number of points that may require manual checking for stricter determination.

Searching for structure in unlabelled data can be challenging because evaluation is largely qualitative. When using K-Means models, choosing K has a large impact on the quality of the analysis; the `KElbowVisualizer` can help select the best K given computational constraints. The `SilhouetteVisualizer` shows the relationship of points in each cluster relative to other clusters and gives an overview of the composition and size of each cluster which may hint at how models group similar data points.

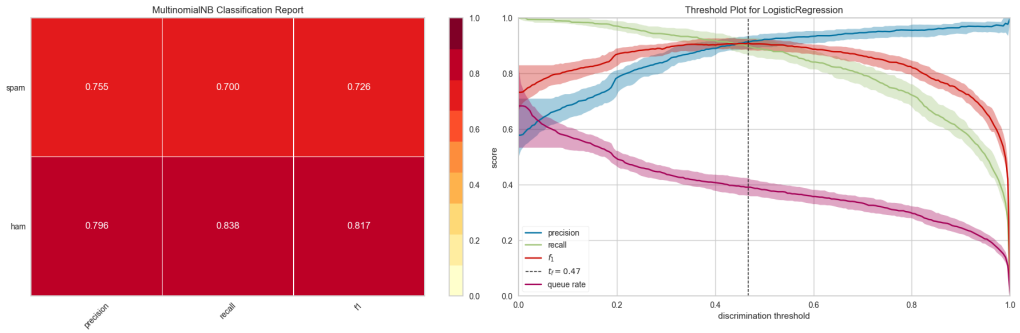

Figure 3: Classification Model Tuning

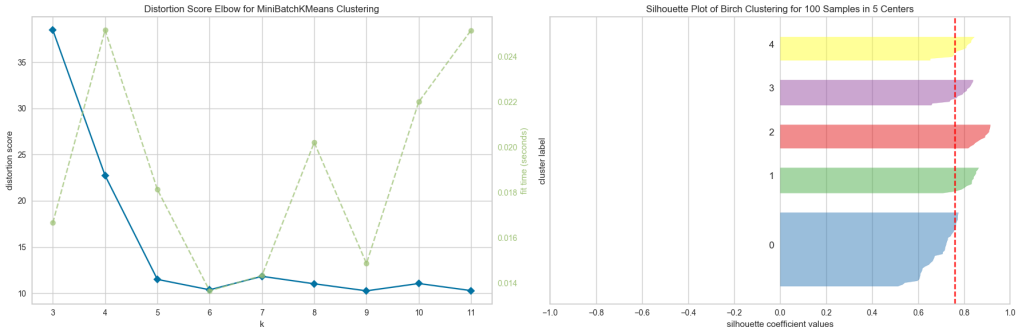

Figure 4: Clustering Model Tuning

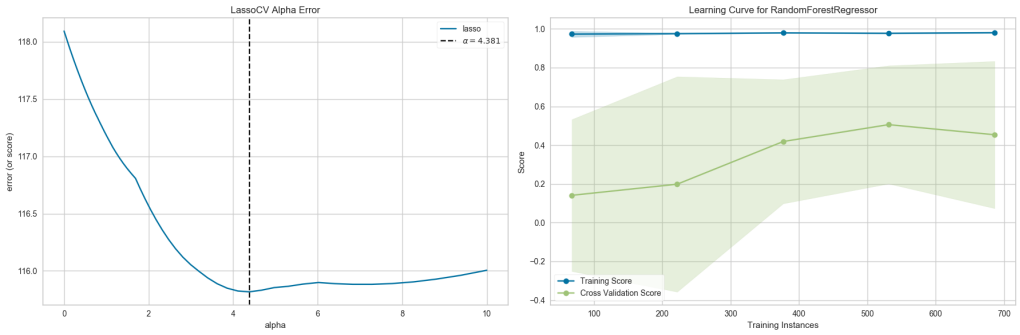

Figure 5: Hyperparameter Tuning

Yellowbrick also offers several other techniques for hyperparameter tuning. Model and regression-specific `AlphaSelection` visualizers help identify the impact of regularization on linear models and the influence of complexity on the trade-off between error due to bias or variance. More generally, the `LearningCurve` visualizer shows how sensitive models are to the amount of data the model is trained on.

Yellowbrick includes many more visualizations, intended to fit directly into the machine learning workflow, and many more are being added in each new release. From text analysis-specific visualizations to missing data analysis, to a `contrib` module that focuses on other machine learning libraries, Yellowbrick has tools to facilitate all parts of hypothesis driven development. The source code for Yellowbrick has been archived to Zenodo and the most recent version can be obtained with the linked DOI: (Bengfort, Bilbro, Danielsen, Gray, & others, 2018).

Acknowledgements

Since we first introduced the idea of Yellowbrick at PyCon 2016, many people have joined us and stuck with us through 12 releases, ensuring the success of the project. Nathan Danielsen joined very early on and was one of our first maintainers, bringing an engineering perspective to our work and giving us much needed stability in testing. Larry Gray, Neal Humphrey, Jason Keung, Prema Roman, Kristen McIntyre, Jessica D'Amico and Adam Morris have also all joined our project as maintainers and core contributors, and we can't thank them enough.

Yellowbrick would not be possible without the invaluable contributions of those in the Python and Data Science communities. At the time of this writing, GitHub reports that 46 contributors have submitted pull requests that have been merged and released, and we expect this number to continue to grow. Every week, users submit feature requests, bug reports, suggestions and questions that allow us to make the software better and more robust. Others write blog posts about using Yellowbrick, encouraging both newcomers and seasoned practitioners to more fully understand the models they are fitting. Our sincere thanks to the community for their ongoing support and participation.

References

Bengfort, B., Bilbro, R., Danielsen, N., Gray, L., & others. (2018). Yellowbrick. doi:10.5281/zenodo.1206239

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, *9*(3), 90–95. doi:10.1109/MCSE.2007.55

Jones, E., Oliphant, T., Peterson, P., & others. (n.d.). SciPy: Open source scientific tools for Python. Retrieved from http://www.scipy.org/

Kapoor, A., Lee, B., Tan, D., & Horvitz, E. (2010). Interactive optimization for steering machine classification. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 1343–1352). ACM. doi:10.1145/1753326.1753529

Kumar, A., McCann, R., Naughton, J., & Patel, J. M. (2016). Model selection management systems: The next frontier of advanced analytics. *ACM SIGMOD Record*, *44*(4), 17–22. doi:10.1145/2935694.2935698

Liu, S., Wang, X., Liu, M., & Zhu, J. (2017). Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, *1*(1), 48–56. doi:10.1016/j.visinf.2017.01.006

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Rajaraman, A. (2008). More data usually beats better algorithms. *Datawocky Blog*.

Wickham, H., Cook, D., & Hofmann, H. (2015). Visualizing statistical models: Removing the blindfold. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *8*(4), 203–225. doi:10.1002/sam.11271