The Journal of Open Source Software

Feature-engine: A Python package for feature engineering for machine learning

Soledad Galli[1]

1 Train in Data

DOI: 10.21105/joss.03642

Software

- Review
- Repository
- Archive

Editor: Øystein Sørensen

Reviewers:

- @Jose-Augusto-C-M
- @papachristoumarios
- @bobturneruk

Submitted: 09 August 2021
Published: 22 September 2021

Summary

Feature-engine is an open source Python library with the most exhaustive battery of transformations to engineer and select features for use in machine learning. Feature-engine supports several techniques to impute missing data, encode categorical variables, transform variables mathematically, perform discretization, remove or censor outliers, and combine variables into new features. Feature-engine also hosts an array of algorithms for feature selection.

The primary goal of Feature-engine is to make commonly used data transformation procedures accessible to researchers and data scientists, focusing on creating user-friendly and intuitive classes, compatible with existing machine learning libraries, like Scikit-learn (Pedregosa et al., 2011) and Pandas (McKinney, 2010).

Many feature transformation techniques learn parameters from data, like the values for imputation or the mappings for encoding. Feature-engine classes learn these parameters from the data and store them in their attributes to transform future data. Feature-engine's transformers preserve Scikit-learn's functionality with the methods fit() and transform() to learn parameters from and then transform data. Feature-engine's transformers can be incorporated into a Scikit-learn Pipeline to streamline data transformation and facilitate model deployment, by allowing the serialization of the entire pipeline in one pickle.

When pre-processing a dataset different feature transformations are applied to different variable groups. Feature-engine classes allow the user to select which variables to transform within each class, therefore, while taking the entire dataframe as input, only the indicated variables are modified. Data pre-processing and feature engineering are commonly done together with data exploration. Feature-engine transformers return dataframes as output, thus, users can continue to leverage the power of Pandas for data analysis and visualization after transforming the data set.

In summary, Feature-engine supports a large variety of commonly used data transformation techniques (Box & Cox, 1964; Dong, 2015; Dror et al., 2011; Kotsiantis et al., 2006; Micci-Barreca, 2001; Yeo & Johnson, 2000), as well as techniques that were developed in data science competitions (Niculescu-Mizil et al., 2009), including those for feature selection (Miller et al., 2009). Thus, Feature-engine builds upon and extends the capabilities of Python's current scientific computing stack and makes accessible transformations that are otherwise not easy to find, understand or code, to data scientist and data practitioners.

Statement of need

Data scientists spend an enormous amount of time on data pre-processing and transformation ahead of training machine learning models (Domingos, 2012). While some feature engineering processes can be domain-specific, a large variety of transformations are commonly applied

across datasets. For example, data scientists need to impute or remove missing values or transform categories into numbers, to train machine learning models using Scikit-learn, the main library for machine learning. Yet, depending on the nature of the variable and the characteristics of the machine learning model, they may need to use different techniques.

Feature-engine gathers the most frequently used data pre-processing techniques, as well as bespoke techniques developed in data science competitions, in a library, from which users can pick and choose the transformation that they need, and use it just like they would use any other Scikit-learn class. As a result, users are spared of manually creating a lot of code, which is often repetitive, as the same procedures are applied to different datasets. In addition, Feature-engine classes are written to production standards, which ensures classes return the expected result, and maximizes reproducibility between research and production environments through version control.

In the last few years, a number of open source Python libraries that support feature engineering techniques have emerged, highlighting the importance of making feature engineering and creation accessible and, as much as possible, automated. Among these, Featuretools (Kanter & Veeramachaneni, 2015) creates features from temporal and relational datasets, tsfresh (Christ et al., 2018) extracts features from time series, Category encoders (McGinnis et al., 2018) supports a comprehensive list of methods to encode categorical variables, and Scikit-learn (Pedregosa et al., 2011) implements a number of data transformation techniques, with the caveat that the transformations are applied to the entire dataset, and the output are NumPy arrays. Feature-engine extends the capabilities of the current Python's scientific computing stack by allowing the application of the transformations to subsets of variables in the dataset, returning dataframes for data exploration, and supporting transformations not currently available in other libraries, like those for outlier censoring or removal, besides additional techniques for discretization and feature selection that were developed by data scientist working in the industry or data science competitions.

Acknowledgements

I would like to acknowledge all of the contributors and users of Feature-engine, who helped with valuable feedback, bug fixes, and additional functionality to further improve the library. A special thanks to Christopher Samiullah for continuous support on code quality and architecture. A list of Feature-engine contributors is available at https://github.com/feature-engine/feature_engine/graphs/contributors.

References

Box, G. E. P., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, *26*(2), 211–243. https://doi.org/10.1111/j.2517-6161.1964.tb00553.x

Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series FeatuRe extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, *307*, 72–77. https://doi.org/10.1016/j.neucom.2018.03.067

Domingos, P. (2012). A few useful things to know about machine learning. *Commun. ACM*, *55*(10), 78–87. https://doi.org/10.1145/2347736.2347755

Dong, Y. (2015). Beating Kaggle the easy way. In *Studienarbeit*. Technische Universität Darmstadt.

Dror, G., Boullé, M., & Guyon, I. (2011). *The 2009 knowledge discovery and data mining competition (KDD cup 2009): Challenges in machine learning*.

Kanter, J. M., & Veeramachaneni, K. (2015). Deep feature synthesis: Towards automating data science endeavors. *2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Paris, France, October 19-21, 2015*, 1–10. https://doi.org/10.1109/DSAA.2015.7344858

Kotsiantis, S., Kanellopoulos, D., & Pintelas, P. (2006). Data preprocessing for supervised learning. *International Journal of Computer Science*, *1*, 111–117.

McGinnis, W. D., Siu, C., S, A., & Huang, H. (2018). Category encoders: A scikit-learn-contrib package of transformers for encoding categorical data. *Journal of Open Source Software*, *3*(21), 501. https://doi.org/10.21105/joss.00501

McKinney, Wes. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

Micci-Barreca, D. (2001). A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *SIGKDD Explor. Newsl.*, *3*(1), 27–32. https://doi.org/10.1145/507533.507538

Miller, H., Clarke, S., Lane, S., Lonie, A., Lazaridis, D., Petrovski, S., & Jones, O. (2009). Predicting customer behaviour: The university of melbourne's KDD cup report. In G. Dror, M. Boullé, I. Guyon, V. Lemaire, & D. Vogel (Eds.), *Proceedings of KDD-cup 2009 competition* (Vol. 7, pp. 45–55). PMLR. http://proceedings.mlr.press/v7/miller09.html

Niculescu-Mizil, A., Perlich, C., Swirszcz, G., Sindhwani, V., Liu, Y., Melville, P., Wang, D., Xiao, J., Hu, J., Singh, M., Shang, W. X., & Zhu, Y. F. (2009). Winning the KDD cup orange challenge with ensemble selection. In G. Dror, M. Boullé, I. Guyon, V. Lemaire, & D. Vogel (Eds.), *Proceedings of KDD-cup 2009 competition* (Vol. 7, pp. 23–34). PMLR. http://proceedings.mlr.press/v7/niculescu09.html

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.

Yeo, I., & Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, *87*(4), 954–959. https://doi.org/10.1093/biomet/87.4.954