The Journal of Open Source Software

Category Encoders: a scikit-learn-contrib package of transformers for encoding categorical data

William D McGinnis [1,2], **Chapman Siu** [3], **Andre Silva** [4], **Hanyu Huang** [5], **Jan Motl** [6], **Kaspar Paul Westenthanner** [7], **and Jonathan Castaldo** [8]

1 Scale Venture Partners **2** Helton Labs, LLC **3** Suncorp Group Ltd. **4** Jungle AI **5** Tencent, Inc. **6** Czech Technical University in Prague **7** Atruvia **8** Meta, Inc.

DOI: 10.21105/joss.10645

Software

- Review
- Repository
- Archive

Editor: Fangzhou Xie

Reviewers:

- @mbsuraj
- @jungtaekkim

Submitted: 01 March 2026
Published: 26 August 2026

Summary

category_encoders is a scikit-learn-contrib module of transformers for encoding categorical data. As a scikit-learn-contrib module, category_encoders is fully compatible with the scikit-learn API (Buitinck et al., 2013). It also makes extensive use of tools provided by scikit-learn (Pedregosa et al., 2011) itself, SciPy (Virtanen et al., 2020), pandas (McKinney, 2010), and statsmodels (Seabold & Perktold, 2010).

The library provides over twenty encoding strategies. These include commonly used encoders such as Ordinal, Hashing, and OneHot (Carey, 2006; "R Library Contrast Coding Systems for Categorical Variables," n.d.; Weinberger et al., 2009), as well as contrast-based encoders including Backward Difference, Helmert, Polynomial, and Sum encoding (Carey, 2006; "R Library Contrast Coding Systems for Categorical Variables," n.d.). It also includes a range of advanced encoders: Target Encoder (Micci-Barreca, 2001), which uses the target variable to derive encodings; CatBoost Encoder (Prokhorenkova et al., 2018), which applies an ordered variant of target encoding to reduce overfitting; Weight of Evidence (WOE) Encoder (Good, 1960), widely used in credit scoring and risk modeling; James-Stein Encoder (James & Stein, 1961), which shrinks estimates toward the overall mean; M-Estimate Encoder (Zadrozny & Elkan, 2002); Generalized Linear Mixed Model (GLMM) Encoder (Bolker et al., 2009); Count Encoder; Quantile Encoder (Micci-Barreca, 2001); Summary Encoder; Gray Encoder; RankHot Encoder; and BaseN Encoder (McGinnis, 2015, 2016; Zhang, 2015).

Statement of need

Categorical variables represent a fixed number of possible values, assigning each observation to one of a finite set of categories. They differ from ordinal variables in that there is no intrinsic ordering among categories. Machine learning algorithms typically require numeric inputs, so categorical data must be converted into a numeric representation before modeling. While simple approaches like one-hot encoding are widely available, many real-world datasets contain high-cardinality categorical features where naive encodings produce excessively wide or uninformative representations.

The original release of category_encoders included a number of commonly used encoders, notably Ordinal, Hashing and OneHot encoders (Carey, 2006; "R Library Contrast Coding Systems for Categorical Variables," n.d.; Weinberger et al., 2009), as well as some less frequently used encoders including Backward Difference, Helmert, Polynomial and Sum encoding (Carey, 2006; "R Library Contrast Coding Systems for Categorical Variables," n.d.). It also included several experimental encoders: LeaveOneOut, Binary and BaseN (McGinnis, 2015, 2016; Zhang, 2015).

Since then, the library has grown substantially through community contributions. It now includes over twenty encoding strategies. Notable additions include Target Encoder (Micci-Barreca, 2001), which uses the target variable to derive encodings; CatBoost Encoder (Prokhorenkova et al., 2018), which applies an ordered variant of target encoding to reduce overfitting; Weight of Evidence (WOE) Encoder (Good, 1960), widely used in credit scoring and risk modeling; James-Stein Encoder (James & Stein, 1961), which shrinks estimates toward the overall mean; M-Estimate Encoder (Zadrozny & Elkan, 2002); Generalized Linear Mixed Model (GLMM) Encoder (Bolker et al., 2009); Count Encoder; Quantile Encoder (Micci-Barreca, 2001); Summary Encoder; Gray Encoder; and RankHot Encoder.

State of the field

Several tools exist for encoding categorical variables, but each covers only a subset of available methods. Scikit-learn (Pedregosa et al., 2011) provides `OrdinalEncoder` and `OneHotEncoder`, but does not include target-based or statistical encoding methods. The pandas (McKinney, 2010) library offers `get_dummies` for one-hot encoding, but without the fit/transform pattern needed for consistent train/test handling. In the R ecosystem, the `recipes` (Kuhn et al., 2024) and `embed` (Hvitfeldt & Kuhn, 2024) packages provide some categorical encoding steps, but with a different API paradigm. The Python libraries Patsy (Smith, 2018) and formulaic (Wardrop, 2023) support contrast coding schemes but are oriented toward formula-based model specification rather than general-purpose feature engineering.

category_encoders fills a gap by consolidating over twenty encoding strategies into a single scikit-learn-compatible package. It is the only Python library that provides supervised encoding methods such as Target, CatBoost, Weight of Evidence, James-Stein, and GLMM encoders alongside classical contrast coding and hashing approaches, all with a uniform interface that handles edge cases like unseen categories and missing values consistently across methods.

Software design

All encoders in category_encoders inherit from a common `BaseEncoder` class that implements the scikit-learn transformer interface (`fit`, `transform`, `fit_transform`). This shared base class provides consistent behavior for column selection, input validation, handling of missing values (via the `handle_missing` parameter), and handling of unseen categories at transform time (via the `handle_unknown` parameter).

Encoders are divided into two families through mixin classes: `UnsupervisedTransformerMixin` for encoders that operate solely on feature values (e.g., Ordinal, OneHot, Hashing, BaseN), and `SupervisedTransformerMixin` for encoders that also use the target variable during fitting (e.g., Target, CatBoost, WOE, James-Stein, GLMM). This design ensures that supervised encoders properly require a target during `fit` and can optionally use it during `transform`, while maintaining API compatibility with scikit-learn's `Pipeline` and model selection utilities.

All encoders accept and return pandas DataFrames, automatically detect categorical columns when none are specified, and support inverse transforms where applicable. The library is designed for production use, with careful handling of edge cases including previously unseen categories, missing values, and invariant columns.

Representative encoding methods

To illustrate the range of techniques the library provides, we give mathematical definitions for a few representative encoders. Consider a single categorical feature that takes values in a set of K categories $\{c_1, \ldots, c_K\}$. Let n_k be the number of training observations in category c_k and $n = \sum_{k=1}^{K} n_k$ the total number of observations.

One-Hot Encoding is the canonical unsupervised scheme: it maps each category c_k to the indicator vector $e_k \in \{0,1\}^K$, the k-th standard basis vector, so that the encoded feature is orthogonal across categories at the cost of a width that grows linearly with cardinality K.

Hashing Encoding keeps the output width fixed regardless of cardinality. Given a hash function h and a chosen number of output dimensions d, category c_k is mapped to bucket $h(c_k) \bmod d$. This bounds the encoded width by d but allows distinct categories to collide into the same bucket.

Target (mean) Encoding is a representative supervised scheme. With a target y, global mean $\bar{y}$, and within-category mean $\bar{y}_k$, category c_k is replaced by a shrinkage estimate that blends the category mean toward the global mean,

$$\hat{x}_k = \lambda(n_k)\,\bar{y}_k + (1-\lambda(n_k))\,\bar{y}, \qquad \lambda(n_k) = \frac{n_k}{n_k + m},$$

where the smoothing parameter $m \geq 0$ controls how strongly low-frequency categories are regularized toward $\bar{y}$. The M-Estimate and James-Stein encoders are variants that differ in how the shrinkage weight λ is chosen.

Weight of Evidence (WOE) Encoding targets binary classification. It encodes category c_k by the log-ratio of the conditional probabilities of the category given each class,

$$\mathrm{WOE}_k = \ln \frac{P(c_k \mid y=1)}{P(c_k \mid y=0)},$$

which is positive when the category is over-represented among positive observations and negative otherwise.

Research impact statement

Since its original release (McGinnis, 2015), category_encoders has been widely adopted across both academic research and applied machine learning. The package has been cited in studies spanning diverse domains, demonstrating its utility as a general-purpose tool for categorical feature engineering.

In machine learning methodology, Poslavskaya and Korolev (2023) used category_encoders to conduct a systematic comparison of encoding methods across OpenML classification benchmarks. In pharmaceutical manufacturing, Schmitt et al. (2022) used the TargetEncoder from category_encoders to encode categorical process parameters when predicting spray-dried dispersion particle size.

These applications — from ML benchmarking to pharmaceutical process modeling — reflect the library's value as a practical, domain-agnostic tool for working with categorical data in Python.

AI usage disclosure

Anthropic's Claude Opus 4.8 (Anthropic, 2026) was used to assist in drafting and editing the text of this paper. No AI tools were used in the development of the category_encoders software.

References

Anthropic. (2026). *Claude Opus 4.8*. https://www.anthropic.com/claude.

Bolker, B. M., Brooks, M. E., Clark, C. J., Geange, S. W., Poulsen, J. R., Stevens, M. H. H., & White, J.-S. S. (2009). Generalized linear mixed models: A practical guide for ecology and evolution. *Trends in Ecology & Evolution*, *24*(3), 127–135. https://doi.org/10.1016/j.tree.2008.10.008

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., Vanderplas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). API design for machine learning software: Experiences from the scikit-learn project. *arXiv Preprint arXiv:1309.0238*. https://doi.org/10.48550/arXiv.1309.0238

Carey, G. (2006). *Coding categorical variables*. https://web.archive.org/web/20220508025941/http://psych.colorado.edu/~carey/Courses/PSYC5741/handouts/Coding%20Categorical%20Variables%202006-03-03.pdf

Good, I. J. (1960). Weight of evidence, corroboration, explanatory power, information and the utility of experiments. *Journal of the Royal Statistical Society: Series B (Methodological)*, *22*(2), 319–331. https://doi.org/10.1111/j.2517-6161.1960.tb00378.x

Hvitfeldt, E., & Kuhn, M. (2024). *Embed: Extra recipes for encoding predictors*. https://github.com/tidymodels/embed

James, W., & Stein, C. (1961). Estimation with quadratic loss. *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, *1*, 361–379.

Kuhn, M., Wickham, H., & Hvitfeldt, E. (2024). *Recipes: Preprocessing and feature engineering steps for modeling*. https://github.com/tidymodels/recipes

McGinnis, W. D. (2015). Beyond one-hot: An exploration of categorical variables. In *Will's Noise*. https://mcginniscommawill.com/posts/2015-11-29-beyond-one-hot-an-exploration-of-categorical-variables/

McGinnis, W. D. (2016). BaseN encoding and grid search in category_encoders. In *Will's Noise*. https://mcginniscommawill.com/posts/2016-12-18-basen-encoding-grid-search-category-encoders/

McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in science conference* (pp. 51–56). 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. *ACM SIGKDD Explorations Newsletter*, *3*(1), 27–32. https://doi.org/10.1145/507533.507538

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, É. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825–2830.

Poslavskaya, E., & Korolev, A. (2023). *Encoding categorical data: Is there yet anything "hotter" than one-hot encoding?* https://doi.org/10.48550/arXiv.2312.16930

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*, 6638–6648. https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html

R library contrast coding systems for categorical variables. (n.d.). In *IDRE Stats*. https://stats.idre.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/

Schmitt, J. M., Baumann, J. M., & Morgen, M. M. (2022). Predicting spray dried dispersion particle size via machine learning regression methods. *Pharmaceutical Research*, *39*(12), 3223–3239. https://doi.org/10.1007/s11095-022-03370-3

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with Python. *9th Python in Science Conference*. https://doi.org/10.25080/majora-92bf1922-011

Smith, N. J. (2018). *Patsy: Describing statistical models in Python*. https://doi.org/10.5281/zenodo.592075

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Vázquez-Baeza, Y. (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

Wardrop, M. (2023). *Formulaic: A high-performance implementation of Wilkinson formulas for Python*. https://github.com/matthewwardrop/formulaic

Weinberger, K., Dasgupta, A., Langford, J., Smola, A., & Attenberg, J. (2009). Feature hashing for large scale multitask learning. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML 09*. https://doi.org/10.1145/1553374.1553516

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 694–699. https://doi.org/10.1145/775047.775151

Zhang, O. (2015). *Strategies to encode categorical variables with many categories*. https://web.archive.org/web/20150926060957/https://www.kaggle.com/c/caterpillar-tube-pricing/discussion/15748