Software Impacts 13 (2022) 100359

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

Categorical Attribute traNsformation Environment (CANE): A python module for categorical to numeric data preprocessing

Luís Miguel Matos [a,*], João Azevedo [c], Arthur Matta [c], André Pilastri [c], Paulo Cortez [a], Rui Mendes [b]

[a] ALGORITMI Centre, Minho University, Guimarães, Portugal
[b] ALGORITMI Centre, Minho University, Braga, Portugal
[c] EPMQ, CCG ZGDV Institute, Guimarães, Portugal

ARTICLE INFO

Keywords:
Data preprocessing
CANE
Python programming language
Machine learning

ABSTRACT

Categorical Attribute traNsformation Environment (CANE) is a simpler but powerful data categorical preprocessing Python package. The package is valuable since there is currently a large range of Machine Learning (ML) algorithms that can only be trained using numerical data (e.g., Deep Learning, Support Vector Machines) and several real-world ML applications are associated with categorical data attributes. Currently, CANE offers three categorical to numeric transformation methods, namely: Percentage Categorical Pruned (PCP), Inverse Document Frequency (IDF) and a simpler One-Hot-Encoding method. Additionally, the CANE module is well documented with several code examples that can help in its adoption by non expert users.

Code metadata

Code metadata description	Information
Current code version	2.2.1.2
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2022-122
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/9329576/tree/v1
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	Python 3.6+
Compilation requirements, operating environments & dependencies	CANE requires bounded-pool-executor; numpy; pandas; pqdm; python-dateutil; pytz; tqdm; typing-extensions; sklearn; and pyspark
If available Link to developer documentation/manual	https://github.com/Metalkiler/Cane-Categorical-Attribute-traNsformation-Environment
Support email for questions	luis.matos@dsi.uminho.pt

1. Categorical attribute transformation environment (CANE)

Currently, Machine Learning (ML) is impacting the world due to the availability of big data (e.g., via digital sensors), computational power and sophisticated algorithms to process such data (e.g., Deep Learning) [1]. Several popular and powerful ML algorithms (e.g., Deep Learning, Support Vector Machines) can only process numerical data. Since real-world applications often generate categorical features, when employing such ML algorithms there is a need to preprocess the data attributes by adopting a categorical to numeric transformation or encoding. Several state-of-the-art ML works (e.g., [2–4]) tend to assume the simpler One-Hot Encoding (1H) method when handling categorical data, which can

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail address: luis.matos@dsi.uminho.pt (L.M. Matos).

https://doi.org/10.1016/j.simpa.2022.100359
Received 29 June 2022; Received in revised form 5 July 2022; Accepted 5 July 2022

produce computational issues in terms of memory and processing effort, particularly when there is a high cardinality [5]. The **Categorical Attribute traNsformation Environment** (CANE) Python module was created to address this issue. CANE offers three simple but effective methods to transform categorical data into numeric values for ML and Deep Learning projects, namely:

- **One-Hot encoding (1H)** — The most popular categorical to numeric transform. The method encodes categorical values into a binary vector with L levels, where L is the number of distinct attribute levels (the cardinality). For instance, the attribute color with $L = 3$ levels {"blue","red","yellow"} would be transformed into: "blue" → (1,0,0); "red" → (0,1,0); and "yellow" → (0,0,1). Since this transform can be applied to virtually any domain, it is the default encoding assumed by most ML tools. One advantage of the CANE implementation of the 1H method is that it allows a fast parallel computing of the transform by encoding each data attribute in a distinct core, thus making use of multi-core machines. Another CANE advantage is that it allows the user to easily name each 1H binary column by using a full name (e.g., ``Blue'', ``Red'') or a suffix (e.g., ``color_blue'' and ``color_red'' for the suffix ``color''). Currently, the 1H transform works with the Pandas Dataframe format.
- **Inverse Document Frequency (IDF)** — proposed by [6] and also explored in [7–9], it is a fast data transformation process that transforms a categorical value into a single numeric value according to the following equation:

$$IDF(t) = ln(\frac{N}{f_t}) \quad (1)$$

where N denotes the total number of instances (values), and f_t is the number of occurrences of level t in the training data. When using this transform, a closer numeric value to 0 means that the level is more frequent in the data. The higher the value, the less frequent the level is, with the less frequent levels being grouped closer. Similar to the 1H method, the IDF CANE implementation allows a multi-core execution (one core for each data attribute). Moreover, the IDF allows to work with two popular data Python formats, the Pandas Dataframe and also Spark Dataframe.
- **Percentage Categorical Pruned (PCP)** — This preprocessing was first introduced in [5] and it works by first sorting the feature levels according to their frequency values. The least frequent levels (summing up to a threshold percentage of P) are merged into a single category denoted as "Others". After this preprocessing method, the one-hot (1H) encoding is applied by using the reduced set of levels, which includes the most frequent levels and the "Others" label. The main goal of the PCP transform is to substantially reduce the input memory and processing requirements while keeping the most relevant levels. In [5], the PCP effect was exemplified by considering a city attribute from a mobile marketing performance task. A small pruning threshold value was adopted ($P = 10\%$), allowing a reduction of 94% in the number of binary inputs (10,690 for 1H and just 689 for PCP). Despite this reduction, the PCP based predictive model (based on a deep learning model) achieved the same predictive performance level when compared with its 1H version but required much less computational effort during the training. Another demonstration is here presented (Fig. 1) for a product quality type from the textile industry. In this example, the vertical dashed line shows the effect of using a threshold of $P = 30\%$, allowing to reduce from 239 (standard 1H) to 40 (PCP) binary inputs (83% in terms of reduction of encoded binary levels). Currently, the PCP transformation uses the Pandas Dataframe format and also includes a multi-core distribution (executing one attribute per core). We highlight that the PCP implementation is exclusive to the CANE Python package.

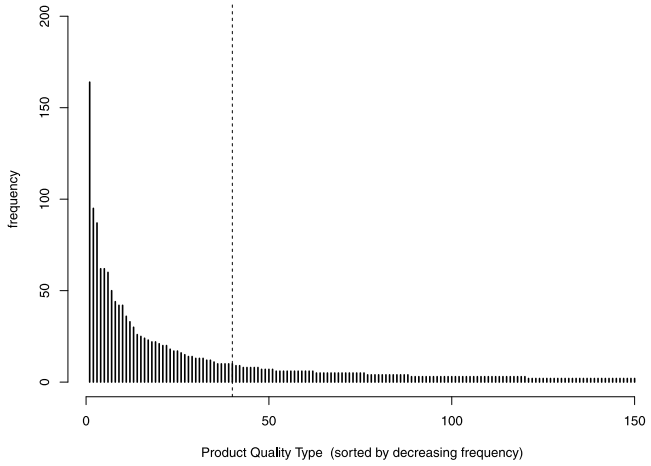

Fig. 1. Example a Product Quality Type attribute reduction that is obtained when using PCP and P = 30% (vertical dashed line).

Table 1
CANE computational execution values (in seconds) when using a synthetic dataset.

Method	Single Core	Multi-core
1H	16.22	24.12
PCP	37.93	22.74
IDF	74.82	31.62

2. CANE impact and computational performance

CANE was used in several scientific studies as a means of reducing the number of inputs (after the categorical to numeric transform) to feed predictive ML models. Diverse real-world applications were addressed, including mobile performance marketing [5,9], Industry 4.0 anomaly detection [4,10] and quality prediction [8,11]. As a consequence of the adoption of CANE python package, the aforementioned studies have observed a reduced computational effort when preprocessing categorical data. Moreover, these transformations were applied to distinct ML algorithms, resulting in more efficient ML implementations (requiring less memory and computational training effort) while keeping high predictive performances. Moreover, the CANE python package,[1] has obtained a total of 76,765 downloads[2] since June 2020.

To demonstrate the computational effort required by the tool, we created a synthetic dataset with 129 attributes and 161,000 records. Each attribute includes 5 distinct levels with different frequencies: "a" – 70,000; "b" – 50,000; "c" – 30,000; "d" – 10,000; and "e" – 1000.[3] The computational experiments assumed the Pandas Dataframe format and were executing using a 2,3 GHz Intel Core i9 machine with a total of 16 cores. Table 1 presents the measured computational effort (in terms of time elapsed) when assuming a pruning threshold value of $P = 5\%$ and two execution scenarios (single core and when using 10 cores). Both PCP and IDF show significant improvements in the computational performance when a multi-core setting is adopted. As for the 1H execution, it is more efficient when adopting a single core setting. This behavior is due to the final multi-core 1H aggregation operation that joins the distinct binary matrices (one for each core) into a single binary Dataframe and that is rather computationally expensive. Nevertheless, we expect that the multi-core 1H Spark version (to be addressed in future work) will produce computationally faster results when compared with its single core version.

[1] Publicly available at https://pypi.org/project/cane.
[2] On June 29th 2022 according to https://pepy.tech/project/cane.
[3] A smaller dataset code demonstration of this example is presented at https://codeocean.com/capsule/9329576/tree/v1.

Table 2
CANE results for a real-world mobile marketing dataset (best values in bold).

Method	Preprocessing time (s)	Training time (s)	# Numeric inputs	Reduction ratio (%)	AUC
1H	163.66	54.98	8,449	0.0	**0.89**
PCP	**10.04**	10.66	**954**	89.7	0.88
IDF	27.70	**8.90**	11	**99.8**	0.73

Another CANE value demonstration example is presented in Table 2, which corresponds to the results that were obtained when using a sample of the data used in [5]. In this example, the dataset contains 10 input categorical features related with a mobile performance marketing domain (e.g., user city). The goal is to predict if a user will buy a product (the conversion result) after seeing a mobile ad (binary classification task). The prediction classifier is based on a Deep Learning model, namely a multilayer perceptron with 9 hidden layers (as described in [5]). Using different testing time periods, three runs were executed on the same computational server (i9 Intel machine, single core execution), with the Deep Learning model being fit with 10,000 training examples and tested with another 5000 instances. In Table 2, the obtained results are presented as the average of the three runs. The 1H encoding results in a very high number of inputs (8449) that substantially affects the preprocessing and ML training time. In contrast, the IDF and PCP (P=10%) CANE methods generate a substantially lower number of inputs, as shown in the columns **# Numeric Inputs** and **Reduction Ratio** (when compared with 1H, in %). This reduction impacts in the computational effort, which is much lower in terms of the preprocessing and training tasks. As for the predictive performance, measured in terms of the Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) curve, quality results were achieved, particularly for the PCP method (only one percentage point lower when compared with the 1H based model).

3. Future work

CANE is a relatively new Python module, having its lifetime spanned across two years. The module was created to solve the issue of transforming high cardinality categorical data into numeric values, which often occurs in real-world applications, such as mobile performance marketing [5]. The main focus was to implement several preprocessing methods (e.g., PCP) that allow non expert ML users to more easily preprocess categorical data attributes to be used in ML applications. The initial version of CANE only handled the PCP and IDF transformations [5,7]. Since then the number of CANE features has increased (e.g., simpler 1H encoding, multi-core implementations, IDF using Spark implementation, Transformation hashmap translations). In the future, we wish to add further capabilities to the module by handing the Spark Dataframe format for the PCP and 1H methods (currently only Pandas Dataframe is addressed). We also aim to optimize the code to be effective and efficient when processing big data datasets (e.g., millions of records with several features), which are common in real-world projects.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors are grateful for project NORTE-01-0247-FEDER-017497, supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF). This work was also supported by FCT Fundação para a Ciência e Tecnologia, Portugal within the Project Scope: UID/CEC/00319/2019. The authors are also grateful for all the contributors that assisted in making CANE more intuitive.

References

[1] Adnan Darwiche, Human-level intelligence or animal-like abilities? Commun. ACM 61 (10) (2018) 56–67, http://dx.doi.org/10.1145/3271625.

[2] Manxing Du, Radu State, Mats Brorsson, Tigran Avanesov, Behavior profiling for mobile advertising, in: Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, BDCAT 2016, Shanghai, China, December 6-9, 2016, 2016, pp. 302–307, http://dx.doi.org/10.1145/3006299.3006339.

[3] Weinan Zhang, Tianming Du, Jun Wang, Deep learning over multi-field categorical data, in: European Conference on Information Retrieval, Springer, 2016, pp. 45–57.

[4] Diogo Ribeiro, Luís Miguel Matos, Guilherme Moreira, André Luiz Pilastri, Paulo Cortez, Isolation forests and deep autoencoders for industrial screw tightening anomaly detection, Comput. 11 (4) (2022) 54, http://dx.doi.org/10.3390/computers11040054.

[5] Luís Miguel Matos, Paulo Cortez, Rui Mendes, Antoine Moreau, Using deep learning for mobile marketing user conversion prediction, in: International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019, IEEE, 2019, pp. 1–8, http://dx.doi.org/10.1109/IJCNN.2019.8851888.

[6] Guilherme O. Campos, Arthur Zimek, Jörg Sander, Ricardo J.G.B. Campello, Barbora Micenková, Erich Schubert, Ira Assent, Michael E. Houle, On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study, Data Min. Knowl. Discov. 30 (4) (2016) 891–927, http://dx.doi.org/10.1007/s10618-015-0444-8.

[7] Luís Miguel Matos, Paulo Cortez, Rui Mendes, Antoine Moreau, A comparison of data-driven approaches for mobile marketing user conversion prediction, in: Ricardo Jardim-Gonçalves, João Pedro Mendonça, Vladimir Jotsov, Maria Marques, João Martins, Robert E. Bierwolf (Eds.), 9th IEEE International Conference on Intelligent Systems, IS 2018, Funchal, Madeira, Portugal, September 25-27, 2018, IEEE, 2018, pp. 140–146, http://dx.doi.org/10.1109/IS.2018.8710472.

[8] Rui Ribeiro, André Pilastri, Carla Moura, Filipe Rodrigues, Rita Rocha, José Morgado, Paulo Cortez, Predicting physical properties of woven fabrics via automated machine learning and textile design and finishing features, in: Ilias Maglogiannis, Lazaros Iliadis, Elias Pimenidis (Eds.), Artificial Intelligence Applications and Innovations, Springer International Publishing, Cham, 2020, pp. 244–255.

[9] Pedro José Pereira, Paulo Cortez, Rui Mendes, Multi-objective grammatical evolution of decision trees for mobile marketing user conversion prediction, Expert Syst. Appl. 168 (2021) 114287, http://dx.doi.org/10.1016/j.eswa.2020.114287, URL https://www.sciencedirect.com/science/article/pii/S0957417420309891.

[10] Gonçalo Fontes, Luís Miguel Matos, Arthur Matta, André Pilastri, Paulo Cortez, An empirical study on anomaly detection algorithms for extremely imbalanced datasets, in: Ilias Maglogiannis, Lazaros Iliadis, John Macintyre, Paulo Cortez (Eds.), Artificial Intelligence Applications and Innovations, Springer International Publishing, Cham, 2022, pp. 85–95.

[11] Luís Miguel Matos, André Domingues, Guilherme Moreira, Paulo Cortez, André Luiz Pilastri, A comparison of machine learning approaches for predicting in-car display production quality, in: Hujun Yin, David Camacho, Peter Tiňo, Richard Allmendinger, Antonio J. Tallón-Ballesteros, Ke Tang, Sung-Bae Cho, Paulo Novais, Susana Nascimento (Eds.), Intelligent Data Engineering and Automated Learning - IDEAL 2021 - 22nd International Conference, IDEAL 2021, Manchester, UK, November 25-27, 2021, Proceedings, in: Lecture Notes in Computer Science, vol. 13113, Springer, 2021, pp. 3–11, http://dx.doi.org/10.1007/978-3-030-91608-4_1.