Proceedings of the 2025 International Conference on Machine Learning and Cybernetics, 13-15 July, 2025

2025 International Conference on Machine Learning and Cybernetics (ICMLC) | 979-8-3315-8736-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICMLC66258.2025.11280100

REPRESENTATION ENHANCEMENT OF TABULAR DATA BY EMBEDDING OF ORDINAL FEATURES

HAN LIU[1], YINGHUI PAN[2], HAO CHEN[3]*

[1]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
[2]School of Artificial Intelligence, Shenzhen University, Shenzhen 518060, China
[3]Information Center, Shenzhen University, Shenzhen 518060, China
E-MAIL: han.liu@szu.edu.cn, panyinghui@szu.edu.cn, haochen@szu.edu.cn
*Corresponding Author

Abstract:

Motivated by the success that transforming symbolic words into numeric vectors through embedding models leads to a significant improvement of feature representation in text processing, in this paper, we propose an approach of numeric encoding to extend the idea of word embedding to general mapping of tabular data. Considering a supervised learning problem where features take symbolic values from ordered space, our method transforms ordinal features into numeric ones within the setting of representation learning. Specifically, our approach aims at transforming label-encoded features into ones represented in a dimensionality-varied space in the setting of multi-scale feature extraction driven by ensemble learning. Experimental results show that our framework indeed achieves enhancing the feature representation, which leads to a significant improvement of learning performance in comparison with widely used encoding methods such as label encoding and feature hashing.

Keywords:

Category encoding; Representation learning; Feature embedding; Ensemble learning

1. Introduction

In machine learning, there are various forms of feature representation, e.g., the original feature representation of image data is in the form of pixels (numeric values of continuous features), whereas text data are originally represented in the form of words, where each word is typically considered as a symbolic feature. In this context, two types of feature representation can be generalized, namely, symbolic description and numeric vectorization.

In real applications, popular learning approaches, such as various types of deep neural networks (DNN), require numeric representation [22, 11, 24, 3, 4]. For example, numeric data like images can be used directly as an input to DNN without any feature pre-processing, but symbolic data like text normally need first to be transformed into a numeric form of representation (e.g., transformation by word embedding approaches [6, 18, 30]) so that the data can be processed by DNN. Also, while a tabular data set contains both symbolic and numeric features, a standard method to handle is to transform symbolic features into numeric ones [20]. On the other hand, a numeric feature is considered to present a larger quantity of information than a symbolic feature. For example, exam results can be presented in the form of either a symbolic feature named as 'Grade' or a numeric one named as 'Mark', but it is obvious that each grade is a qualitative representation of a range of marks, i.e., it is a many-to-one mapping from marks to grades. Moreover, the above-mentioned symbolic feature can be provided only with the frequency distribution among various grades, but more useful information on the distribution of marks within each grade can be obtained using numeric feature representation. The above argumentation indicates the necessity to transform symbolic features into numeric ones, in order to suit those popular learning approaches and increase the quantity of useful information presented in the feature space towards maximizing the correlation between features and labels.

In general, symbolic features can be specialized further into various types, e.g., nominal, ordinal and string [8]. The difference between nominal and ordinal features is that there is a sequential relationship among the values of an ordinal feature whereas the values of a nominal attribute are not ordered [31]. In terms of feature cardinality, a nominal or ordinal feature

979-8-3315-8736-9/25/$31.00 ©2025 IEEE

needs to involve a finite number of predefined symbolic values, whereas the domain of a string attribute is made up of an infinite number of symbolic values. The focus of this paper is on the embedding of ordinal features in tabular data, while the domain of each feature is assumed to consist of a small number of predefined values (occurring in both training and test sets) and the class attribute is set to be nominal. The above assumptions of data characteristics may hold in some scenarios such as medical analysis [23] and nursing [16].

In this paper, we propose a theoretical framework for numeric encoding of ordinal features. Also, the proposed framework is designed within the setting of representation learning, in contrast to traditional methods that typically encode in hand-crated ways and treat the feature transformation as a preprocessing task [22]. The contributions of this paper include the following:

- We have suggested a feature virtualization strategy for transforming label-encoded features into ones represented in a new space with varied dimensionality.
- The experimental results show that the performance of feature representation is improved considerably through adopting the proposed framework in comparison with widely adopted encoding methods.

2 Proposed Approach of Numeric Encoding of Ordinal Features

Based on the numerical features obtained by using label encoding to transform each symbolic value of an ordinal feature into a number, we propose a representation learning approach referred to as feature virtualization. Specifically, feature virtualization is a supervised operation, which is designed to transform the label-encoded features into ones represented in a new space with varied dimensionality in the setting of joint training of features and predictive models. The significance of the supervised operation is at producing a virtual feature space that consists of multiple subspaces that are all identical to the label space, towards relevance maximization and redundancy minimization of features. The operation is driven by training multiple regression ensembles and concatenating the outputs of these ensembles to obtain a virtual feature vector for each instance. Moreover, the suitable setting of the number of ensembles generally varies for different scales of data, so the dimensionality of the newly created virtual feature space is set to be varied.

The entire procedure of feature virtualization is illustrated in Algorithm 1, where feature sub-sampling is adopted to obtain M subsets $\{Sub_{bias}^{(s)}\}_{s=1}^{M}$ of FS_{bias} and each subset $Sub_{bias}^{(s)}$ is used to train k regression ensembles $\{ens_{st}\}_{t=1}^{k}$ to predict numeric values for k pre-defined classes in the setting of gradient boosting [10]. Finally, $M \times k$ virtual features are produced, which each is obtained by normalizing the output of a regression ensemble. The normalization of each output can be achieved by using a softmax function as shown in Eq. (1):

$$P_t^{(s)} = \frac{exp(\bar{v}_t^{(s)})}{\sum_{\bar{t}=1}^{k} exp(\bar{v}_{\bar{t}}^{(s)})}, \tag{1}$$

where $\bar{v}_t^{(s)}$ denotes the output of regression ensemble ens_{st} trained using the s-th feature subset $Sub_{bias}^{(s)}$ for the t-th class c_t and $P_t^{(s)}$ is the normalized output.

The necessity of normalizing the output of each regression ensemble lies in two aspects. Firstly, the output may be a negative value due to the case that the regression ensemble is constructed by learning each base model reg_h to fit the residual r of the additive regression ensemble involving previous models $reg_1, reg_2, \ldots, reg_{h-1}$, as shown in Eq. (2):

Algorithm 1: Feature virtualization procedure

Input : a set of label-encoded features FS_{bias}, a training set TR, each instance $X_e \in TR$
Output: a set of virtual feature vectors VS_{vir}

for $s = 1$ *to* M **do**
  set the feature subspace size $size_s$;
  randomly select $size_s$ features from FS_{bias};
  for $t = 1$ *to* k **do**
    train a regression ensemble ens_{st} for class c_t, using $size_s$ selected features;
    for $e = 1$ *to* $|TR|$ **do**
      ens_{st} outputs a numeric value $\bar{v}_{et}^{(s)}$ of class c_t for instance X_e;
      add $\bar{v}_{et}^{(s)}$ as the value of the t-th element of class vector $vec_e^{(s)}$ of X_e;
      if $t = k$ **then**
        $\overline{vec}_e^{(s)} = Softmax(vec_e^{(s)})$;
        add $\overline{vec}_e^{(s)}$ into vector set $VS_{vir}^{(s)}$;
      end
    end
  end
end
generate VS_{vir} by concatenating horizontally the vector sets $VS_{vir}^{(1)}, VS_{vir}^{(2)}, \ldots, VS_{vir}^{(M)}$;

$$r = Y - \sum_{\bar{h}=1}^{h-1} reg_{\bar{h}}(X), \tag{2}$$

where Y denotes the true output (0 or 1), r is ranged in $[-1, 1]$, and $\sum_{\bar{h}=1}^{h-1} reg_{\bar{h}}(X)$ represents the output of a regression ensemble $\{reg_{\bar{h}}\}_{\bar{h}=1}^{h-1}$, given an input X.

While the output of a regression ensemble is negative, the exponential function shown in Eq. (1) can map the negative value into a positive one. Also, it is necessary to ensure that the outputs of the k ensembles $ens_{s1}, ens_{s2}, \ldots, ens_{sk}$ (trained for k classes using the same feature subset $Sub_{bias}^{(s)}$) are all ranged in $[0, 1]$ and can be summed to 1, such that the normalized outputs can be concatenated as a k-dimensional class vector $vec_e^{(s)}$ of instance $x_e \in c_t$ and the difference between $vec_e^{(s)}$ and the one-hot vector o_t of class c_t can be measured more effectively. In this context, if the above-mentioned difference is small for all instances, then the correlation between features and labels is said to be high, where the class vector of each instance $x_e \in c_t$ is an input and the one-hot vector of c_t is the expected output. In our setting of feature virtualization, the maximization of the aforementioned correlation is achieved indirectly by minimizing the mean squared error (MSE) of each ensemble ens_{st}, i.e., if all the k ensembles, which are trained for k classes using the same feature subset, have low MSE, a small difference between $vec_e^{(s)}$ of each instance $x_e \in c_t$ and o_t of c_t is obtained, which is equivalent to a low cross entropy and high correlation between features and labels.

The second aspect concerning the necessity of normalization is thought as the usefulness of Softmax in reducing the impact of a regression error on the above-mentioned loss, according to Eq. (3) that illustrates the partial derivative of the Softmax function with respect to the regression output $\bar{v}_{\bar{t}}^{(s)}$ of each ensemble ens_{st}.

$$\frac{\partial P_t^{(s)}}{\partial \bar{v}_{\bar{t}}^{(s)}} = \left\{ \begin{array}{ll} P_t^{(s)}(1 - P_{\bar{t}}^{(s)}) & t = \bar{t}; \\ -P_t^{(s)} P_{\bar{t}}^{(s)} & t \neq \bar{t}; \end{array} \right\}, \tag{3}$$

Specifically, the gradient of each normalized output $P_t^{(s)}$ with respect to the regression output $\bar{v}_{\bar{t}}^{(s)}$ is ranged in $(-1, 0)$ or $(0, 1)$, which indicates that the increase or decrease of $P_t^{(s)}$ is slower than the change of the corresponding regression output $\bar{v}_t^{(s)}$ for class c_t. Since the cross entropy is only affected by the probability estimation of the ground truth label, the impact of a regression error on the cross entropy can be reduced after the normalization.

Moreover, in order to achieve independent learning of each virtual feature, we set to train k regression ensembles separately for k pre-defined classes, and normalization of these ensemble outputs is taken only after the completion of training the ensembles, unlike the classification-oriented gradient boosting strategy that needs to normalize the k ensemble outputs at the end of each iteration of training base models. Also, inspired from the idea of the cascading residual network-based representation learning [17], we design to involve residual-driven training of additive regression ensembles towards generating high quality features.

In addition, the adoption of feature sub-sampling at the beginning of Algorithm 1 is considered as a strategy of overfitting avoidance, i.e., since feature subsampling has been adopted in the designs of those popular ensemble approaches (e.g., random subspace [15] and random forests [1]) as an effective strategy of reduction of prediction variance, the strategy is considered to be useful in reducing the variability of feature representation for each instance by avoiding high variance of regression ensembles. The setting to produce multi-sized feature subsets by sub-sampling is made due to the consideration that the number of useful features may vary for different instances, e.g., the leaf nodes of a decision tree may be at different levels, which indicates that some instances can be classified by using fewer features but classifying other instances needs more features. Therefore, the above setting is considered as a kind of multi-scale feature extraction [12, 19], considering that new features are transformed from various-dimensional feature subsets.

Overall, the key point of feature virtualization is to transform label-encoded features into $M \times k$ virtual features independently, i.e., M subsets of numeric features, which are all subsampled from a set of label-encoded features, are transformed into M groups of new features, where each group involves k features that are obtained by passing the outputs of k regression ensembles that are trained on a subset of label-encoded features for estimating k class probabilities. It is worth noting that stacking [29] involves a similar operation, i.e., passing outputs of models as features to a further learner. However, feature virtualization is designed to perform representation learning whereas stacking is essentially to train a combiner to fuse the outputs of multiple models.

3 Experiment

In this section, we report our experimental study by specifying its setup and discussing the results to analyse the effectiveness and limitations of our proposed framework.

In this experimental study, we select 10 data sets from the

UCI repository [7] and the Weka website [9]. The details of these data sets are shown in Table 1.

TABLE 1. Characteristics of data sets

Dataset	No.of ordinal features	No.of instances	No.of classes
Balance	4	625	3
Breast	6	278	2
Car	6	1728	4
CMC	7	1473	3
ERA	4	1000	9
ESL	4	488	9
LEV	4	1000	5
Nursery	8	12960	5
SWD	10	1000	4
Wisconsin	9	683	2

[1] 'CMC' dataset contains two numeric features ('wife's age' and 'number of children ever born'), and the other datasets have no numeric feature.
[2] Three nominal features ('menopause', 'breast', 'breast-quad') in the 'Breast' dataset are deleted, and a nominal feature 'id' in the 'Wisconsin' dataset is removed.
[3] 'Breast' and 'Wisconsin' datasets have some instances with missing values, and such instances are removed, so the number of instances without missing values is specified in the table for each of the two data sets.
[4] The standard names of the 'Balance', "Breast' and "Wisconsin' data sets are Balance Scale, Breast Cancer, Wisconsin Breast Cancer (Original), respectively.

The experiment on each data set is conducted through one run of 10-fold cross validation, where area under curve (AUC) is selected as the evaluation metric.

The baseline methods selected for comparison with our proposed framework include binary encoding (BE) [13], count encoding (CE) [13], hash encoding (HE) [28], one-hot encoding (OE) [5], label encoding (LE) [3], generalized linear mixtual models (GLMM) [5], M-estimator (ME) [22], target encoding (TE) [22], weight of evidence (WOE) [13] and ordered target statistics (OTE) [27]. In terms of the implementation of the experiment, two Python libraries, namely, *Scikit-learn* 0.20.1 [26] and *Category Encoders* 2.2.2 [21], are used. In particular, those aforementioned baseline methods are available in *Category Encoders*, and the implementation of feature virtualization is based on *Scikit-learn*.

In terms of hyper-parameter selection, we adopt the tuned settings of hyper-parameters for those existing encoding methods (that have at least one hyper-parameter). Specifically, for each of these methods (that have at least one hyper-parameter), the hyper-parameters are jointly tuned by selecting the option (hyper-parameter vector) that leads to the best performance of 5-fold cross validation on the training set. The strategies of hyper-parameters tuning are specified in Table 2.

Moreover, feature virtualization has some hyper-parameters relating to feature sub-sampling and gradient boosting. Specifically, we set to tune the number of iterations (n_iterations) and the sub-sampling proportions in various iterations (proportions) for feature sub-sampling. Also, the classification and regression tree (CART) algorithm [2] is used to train each base model in the setting of gradient boosting, where three

TABLE 2. Hyper-parameters tuning strategies for encoding methods

Method	Hyper-parameter	List of options
HE	hash_method	['md5', 'sha256', 'black2b']
	n_components	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
TE	smoothing	[0.1, 0.2, 0.5, 1.0, 2.0, 5.0]
ME	m	[0.1, 0.2, 0.5, 1.0, 2.0, 5.0]
WOE	regularization	[0.1, 0.2, 0.5, 1.0, 2.0, 5.0]
OTS	sigma	[0.1, 0.2, 0.5, 1.0, 2.0, 5.0]
	a	[0.1, 0.2, 0.5, 1.0, 2.0, 5.0]

hyper-parameters are tuned, namely, the maximum tree depth (max_depth), the learning rate (learning_rate) and the maximum number of iterations (n_estimators) for gradient boosting. The selection of the aforementioned hyper-parameters is driven by conducting 5-fold cross validation on the training set and picking up the settings that lead to the best performance, whereas we keep the default settings (provided in *Scikit-learn*) for all the other hyper-parameters of CART and gradient boosting.

The strategies of tuning the hyper-parameters involved in the proposed approach are specified in Table 3. In particular, the three hyper-parameters of gradient boosting are jointly tuned first and then the optimized settings of the three hyper-parameters are kept as the basis for jointly tuning the two hyper-parameters of feature sub-sampling.

TABLE 3. Strategies of hyper-parameters tuning for proposed approach

Method	Hyper-parameter	List of options
Gradient Boosting Regressor	learning_rate	[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1]
	n_estimators	[1, 3, 5, 7, 9, 30, 50, 70, 90, 100]
	max_depth	[1, 2, 3, 4, 5]
Feature Sub-sampling	n_iterations	[1, 3, 5]
	proportions	[1.0], if n_iterations=1
		[0.6, 0.8, 1.0], if n_iterations=3
		[0.6, 0.7, 0.8, 0.9, 1.0], if n_iterations=5

For all of the encoding methods, their performance of feature transformation is finally evaluated by measuring the AUC of classifiers trained by SVM, where the default hyper-parameter settings (provided in *Scikit-learn*) for SVM are kept to train classifiers using all those numeric feature sets produced by various encoding methods.

The comparison of the proposed approach with those unsupervised encoding methods is shown in Table 4, whereas Table 5 shows the performance comparison between the proposed approach and each of those supervised encoding methods. The results generally indicate that the proposed approach outperforms those existing methods of feature encoding in all cases.

The main reason why combining label encoding and feature virtualization is helpful to advance the performance is thought to be the supervised setting of joint training of features and regression models. In this setting, each subset of the numeric

TABLE 4. Comparison with various unsupervised methods

Dataset	Binary	Count	Hashing	OneHot	Label	Proposed
Balance	0.8873	0.6594	0.8036	0.8742	0.9543	**0.9818**
Breast	0.6951	0.6868	0.7622	0.7190	0.6694	**0.8533**
Car	0.9860	0.7458	0.8371	0.9981	0.9961	**0.9996**
CMC	0.7047	0.6191	0.7073	0.7120	0.7091	**0.7408**
ERA	0.7475	0.7275	0.7426	0.7459	0.6805	**0.7608**
ESL	0.9110	0.8348	0.8961	0.9111	0.8594	**0.9387**
LEV	0.8450	0.7487	0.8164	0.8443	0.8239	**0.8632**
Nursery	0.9953	0.7621	0.9865	0.9989	0.9937	**0.9994**
SWD	0.8063	0.7678	0.7919	0.7998	0.7894	**0.8316**
Wisconsin	0.9924	0.9836	0.9934	0.9913	0.9842	**0.9952**

TABLE 5. Comparison with various supervised methods

Dataset	GLMM	M-Estimator	Target	WOE	OTS	Proposed
Balance	0.9473	0.9452	0.9449	0.9474	0.5612	**0.9818**
Breast	0.6413	0.6810	0.6792	0.6783	0.4600	**0.8533**
Car	0.9794	0.9890	0.9890	0.9865	0.9515	**0.9996**
CMC	0.7128	0.7076	0.7076	0.7131	0.7107	**0.7408**
ERA	0.7392	0.7441	0.7439	0.7474	0.7258	**0.7608**
ESL	0.9168	0.9107	0.9103	0.9205	0.9116	**0.9387**
LEV	0.8443	0.8435	0.8437	0.8475	0.8378	**0.8632**
Nursery	0.9522	0.9814	0.9814	0.9823	0.8283	**0.9994**
SWD	0.7997	0.8020	0.8020	0.7998	0.8005	**0.8316**
Wisconsin	0.9921	0.9896	0.9898	0.9924	0.9831	**0.9952**

feature set produced by label encoding that maps each categorical value of an ordinal feature into a specific number is transformed into a subset of virtual features that form a feature subspace that is approximately identical to the label space, so the performance of the regression models is positively correlated to the quality of the features generated at the feature virtualization stage, i.e., while k regression models are learned to output k numeric values that are normalized by softmax and are then concatenated as a feature vector for representing each instance, highly accurate outputs of the models would be transformed into such a vector that is very close to the one-hot vector of the ground truth label of the instance. Moreover, feature virtualization is designed by taking into account the bias-variance trade-off [14], where the bias reduction is achieved by involving residual-driven ensemble learning and feature sub-sampling is adopted for the variance reduction. Therefore, it can be expected to generate stably a feature vector that is close to the one-hot vector of the ground truth label for each instance.

4. Conclusions

In this paper, we have proposed an approach of numeric embedding of ordinal features within the setting of representation learning. In particular, the proposed approach leads to enhancing the feature representation by producing a dimensionality-varied space, towards maximizing the correlation between features and labels. The experimental results show that the performance is improved considerably through adopting the proposed approach for enhancing the feature representation, in comparison with widely used encoding methods. In the future, we will investigate how to map each symbolic value into a numeric interval that follows a normal distribution. It is also worthy of further studies to address the aforementioned limitations of the proposed framework in the context of open-world learning [25]. Moreover, in the big data era, it has been a commonly accepted point that the learning needs to go deeper [33, 32], which means that the model needs to have a sufficient complexity in order to be able to contain comprehensive knowledge to improve the generalization performance. Therefore, it is also worth to extend our proposed framework towards performing a more complex procedure of representation enhancement in the setting of deep learning.

Acknowledgements

This paper is supported by National Natural Science Foundation of China (Grants 62106147 and 62276168), Guangdong Provincial Natural Science Foundation (Grant 2023A1515010869).

References

[1] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall/CRC, Monterey, CA, 1984.

[3] R. K. Brouwer. A feed-forward network for input that is both categorical and quantitative. *Neural Networks*, 15:881–890, 2002.

[4] P. Cerda and G. Varoquaux. Encoding high-cardinality string categorical variables. *IEEE Transactions on Knowledge and Data Engineering*, 34(3):1164–1176, 2022.

[5] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken. *Applied Multiple Regression/Correlation Analysis For The Behavioral Sciences, 3rd ed*. Routledge, London, 2003.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In M. Walker, H. Ji, and A. Stent, editors, *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics*, New Orleans, Louisiana, 2018. Association for Computational Linguistics.

[7] D. Dua and C. Graff. UCI machine learning repository, 2017.

[8] E. Frank, M. A. Hall, and I. H. Witten. *The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques, 4th ed*. Morgan Kaufmann, San Francisco, 2016.

[9] E. Frank, M. A. Hall, and I. H. Witten. Weka: The workbench for machine learning, 2021.

[10] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[11] K. Grabczewski and N. Jankowski. Transformations of symbolic data for continuous data oriented models. In *Proceedings of the Joint International Conference on Artificial Neural Networks and Neural Information Processing*, pages 359–366, Istanbul, Turkey, 26-29 June 2003. Springer.

[12] C. Guo, B. Fan, Q. Zhang, S. Xiang, and C. Pan. Augfpn: Improving multi-scale feature learning for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2020.

[13] J. T. Hancock and T. M. Khoshgoftaar. Survey on categorical data for neural networks. *Journal of Big Data*, 7(1), 2020.

[14] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, London, 2009.

[15] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[16] U. Jakobsson. Statistical presentation and analysis of ordinal data in nursing research. *Scandinavian Journal of Caring Sciences*, 18(4):437–440, 2004.

[17] R. Lan, L. Sun, Z. Liu, H. Lu, Z. Su, C. Pang, and X. Luo. Cascading and enhanced residual networks for accurate single-image super-resolution. *IEEE Transactions on Cybernetics*, 51(1):115–125, 2021.

[18] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, pages II–1188–II–1196, Beijing, China, 21-26 June 2014. Springer.

[19] G. Li and Y. Yu. Visual saliency based on multiscale deep features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5455–5463, 2015.

[20] Y. Li, X. Fan, and E. Gaussier. Supervised categorical metric learning with schatten p-norms. *IEEE Transactions on Cybernetics*, 52(4):2059–2069, 2022.

[21] W. D. McGinnis, C. Siu, A. S, and H. Huang. Category encoders: a scikit-learn-contrib package of transformers for encoding categorical data. *Journal of Open Source Software*, 3(21):501, 2018.

[22] D. Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD Explorations Newsletter*, 3(1):27–32, 2001.

[23] H. A. Miot. Analysis of ordinal data in clinical and experimental studies. *Jornal Vascular Brasileiro*, 19, 2020.

[24] K. G. Nski and G. Stawski. Symbolic features in neural networks. In *Proceedings of the 5th Conference on Neural Networks and Their Applications*, 2000.

[25] J. Parmar, S. S. Chouhan, V. Raychoudhury, and S. S. Rathore. Open-world machine learning: Applications, challenges, and opportunities. *ACM Computing Surveys*, 2023.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. T. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[27] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, 2018.

[28] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120, Montreal, Canada, 14-18 June 2009. ACM.

[29] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[30] O. Wu, T. Yang, M. Li, and M. Li. Two-level lstm for sentiment analysis with lexicon embedding and polar flipping. *IEEE Transactions on Cybernetics*, 52(5):3867–3879, 2022.

[31] Y. Zhang and Y.-M. Cheung. A new distance metric exploiting heterogeneous interattribute relationship for ordinal-and-nominal-attribute data clustering. *IEEE Transactions on Cybernetics*, 52(2):758–771, 2022.

[32] Z. Zhou and J. Feng. Deep forest. *National Science Review*, 6:74–86, 2019.

[33] Z.-H. Zhou and J. Feng. Deep forest: Towards an alternative to deep neural networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3553–3559, 02 2017.