International Journal of Computer Applications (0975 – 8887)
Volume 186 – No.44, October 2024

Cat2Vec with Position Encoding: A New Approach for Handling Ordinal Features using Learned Embeddings with Positional Encoding

Aditya Narvekar, PhD
SP Jain School of Management
15 Carter Street,
Sydney, NSW, Australia

Shubh Mehta
SP Jain School of Management
15 Carter Street,
Sydney, NSW, Australia

ABSTRACT
Machine learning projects spend a significant amount of their time and money on pre-processing data. Often the success of Machine learning projects depends on how the features are handled and processed before model building begins. Without rigorous exploration and preprocessing of features machine learning projects will often suffer from time and cost overruns. This study proposes a new technique called “cat2vec with position” to handle categorial features. For nominal features this study proposes the use learned embeddings. This study proposes a new technique that uses learned embeddings with positional encoding for ordinal features. Position encoding is a technique used with transformers to encode relative position of words in a sentence. This study adapts this technique for ordinal variables. Ordinal variables are categorical variables whose values have an inherent position. The authors wrote the code for learning position encodings for ordinal variables. This study used a large dataset which contained a mix of nominal and ordinal variables to run experiments. Experiments were based on sklearn pipelines where each pipeline covered an approach to preprocessing. Pipelines were built using the typical approach, the new approach, as well as hybrid pipelines that combine elements of both the traditional and the new approach. The experiments demonstrate that the new approach, named “Cat2Vec with position,” outperforms traditional techniques for handling nominal and ordinal variables. To the best of current knowledge, this is the first study to apply a positional encoding technique from NLP to encode ordinal variables.

General Terms
Machine learning, learned embeddings, learned embeddings with positions, categorical variable, nominal variable, ordinal variable, linear regression, k-nearest neighbors, random forests, support vector machines, XGBoost, positional encoding, big data

Keywords
Machine learning, learned embeddings, learned embeddings with positions, categorical variable, nominal variable, ordinal variable, linear regression, k-nearest neighbors, random forests, support vector machines, XGBoost, positional encoding, big data

1. INTRODUCTION
Machine learning projects involve a significant amount of exploratory data analysis and data pre-processing. Studies indicate the significant impact of data pre-processing on not only the performance of the models built but also on the cost of running the machine learning project [1][2][25]. Studies have also looked at various aspects such as the effect of normalization, discretization and dimensionality reduction on the performance of machine learning models [3]. This study focuses on the techniques related to handling categorial variables. Categorical variables can be further divided into nominal and ordinal variables. Nominal variables are categorical variables that do not have inherent ranking between the values. For example, color could be red, blue, green but there is no ranking to these values. On the other hand, ordinal variables are also categorical variables, but the values of an ordinal variable have a natural order or ranking. For example, reviews be labeled as good, neural or bad. These values have a ranking where good can be considered closer to neutral than to bad. The author’s find this ordering like the ordering of words in the sentence. Natural language processing (NLP) has made significant strides in the recent years. One of the key innovations in the field of NLP was the creation of transformers. The paper that proposed use of attention with transformers also proposed the use of position encoding using sine and cosine functions of varying frequencies [4]. This study takes the idea of positional encoding from NLP and combines it with column transformations which can be used by anyone using sklearn pipelines. Further, this study, takes a large dataset from Kaggle named “IT_Incident_log_dataset” which contains a total of 141,712 samples. This dataset was chosen because it contains a mixture of nominal and ordinal variables which we can use to study the improvement offered by the new approach proposed by this study.

1.1 Significance of this study
Several past studies have looked at techniques used to encode categorical variables [5][6][7][8][9][10]. These studies have mostly focused on different techniques for finding new representations of nominal variables because the standard one-hot encoding technique leads to an increase in the number features in the dataset. Ordinal features are mostly encoded using label encoding which assigns a number to each category. This has worked well because the straightforward numbering process for label encoding also encodes the position by default. However, for high cardinality ordinality features, the distance between label encoded values is likely to be excessive and could lead to machine learning models learning unwanted patterns that do not represent the real word. This study for the first time uses learned embeddings for both nominal and ordinal variables using a deep neural network. Furthermore, for ordinal variables, positional encoding is added using sine and cosine functions of varying frequency, a method originally proposed by NLP researchers. [4]. This study then uses a large dataset to apply the new technique and then compares its performance with the previous encoding techniques.
The rest of the paper is structured as follows –

Section 2 – Describes the existing literature specifically used for encoding nominal and ordinal features.
Section 3 – Describes the data and methodology used to develop the new technique named "Cat2vec with position encoding" proposed by this study
Section 4 – Describes the results of the experiments done to compare the performance of our new approach.
Section 5 – Presents our final comments and discusses the avenues for further research.

2. LITERATURE REVIEW

Data pre-processing has been studies for several years. Most of the research has focused on reviewing existing processes and coming up steps to create a standard process for data pre-processing before model building. However, these studies tend to focus on the process rather than on the techniques used [11][12][13][14][18]. Similarly, some studies have focused on data-processing processing techniques used for a particular problem such as early diagnosis of diabetes, heart and liver diseases [12] or for house price index prediction problem [13] or for heart disease prediction [16]. Also, most studies have used the same standard pre-processing techniques for categorical variables. They have mostly used one-hot encoding for nominal variables and label encoding for ordinal variables. Some studies apply specialized neural networks for learning embeddings for categorical variables having high cardinality [15] but they treat do not make any distinction between nominal and ordinal features. Researchers have also applied specialized techniques such as binary decomposition for handling data with some degree of uncertainty [17]. To summarize, these studies have largely recommended processes and guidelines to follow for data pre-processing but have done limited work on researching newer innovative techniques that can be used to create new representations of nominal and ordinal variables.

Researchers have worked on using convolutional neural networks for learning embeddings for ordinal variables instead of using one-hot encoding for DNA motif discovery [19]. The use of one-hot encoding for ordinal variables is not ideal because one-hot encoding do not maintain the order between values of an ordinal feature. Ordinal encoding was also used in work related to prediction of anticancer peptides [20]. However, this study does not create position encoding using sine and cosine functions of varying frequencies like this research does. Another paper proposes use of complex encodings that provide a symmetric representation of categorical values in the complex plane [21]. The complex encoding proposed by this study does not differentiate between nominal and ordinal variables. The authors of this study believe that the inherent order contained in the values of ordinal variables represents important information that supervised learning algorithms should learn. Some studies do not differentiate between nominal and ordinal variables and have proposed techniques that use deep neural networks to learn embeddings for both ordinal and nominal features [21][22]. While use of learned encodings seems a promising approach based on the results of previous studies that have reported superior performance as compared to traditional methods [19][20][21][22], the lack of differentiation between ordinal and nominal features will lead to loss of information related to inherent order in ordinal features. This study therefore differentiates itself from previous studies not only by learning deep embeddings for both nominal and ordinal variables, but also by adding positional encoding for ordinal variables. Our position encoding technique is based on positional encoding technique used with transformers in the popular large language models (LLM) [4][23][24].

3. DATA AND METHODOLOGY

This study is based on a dataset taken from www.kaggle.com. For this study, a regression problem was selected. There are plans to apply the technique to classification problems, with results to be published in a follow-up study. A real-world big data set was selected to compare the performance of the new "Cat2Vec with position encoding" technique with existing techniques.

3.1 Data set

Several datasets were considered, and the decision was made to use the "IT_Incident_log_dataset". This dataset is constructed from an event log of an incident management process, extracted from data gathered from the audit system of a ServiceNow™ platform instance utilized by an IT company. The event log is enriched with data loaded from a relational database underlying a corresponding process-aware information system. Information was anonymized for privacy.

This dataset will be referred to as the "incident" dataset from this point forward. The "incident" dataset consists of 141,712 samples and 36 columns. This dataset was chosen due to its favourable mixture of ordinal, nominal, and continuous variables. Exploratory data analysis was conducted, and the features were classified as shown in Table 1.

Table 1. Classification of features of Incident Dataset

Ordinal variables	Nominal variables	Continuous variables	Processed to continuous variable
incident_state	active	reassignment_count	sys_updated_by
impact	made_sla	reopen_count	location
priority	contact_type	sys_mod_count	category
u_priority_confirmation	knowledge	sys_updated_at	subcategory
notify			closed_code
			resolved_by

As shown in Table 1, this dataset was selected due to the presence of 6 ordinal variables and 4 nominal variables, thereby enabling the application of the new technique that is being proposed by this study.

The dependent variable in the incident dataset were closed_at and resolved_at which represent the date and time when the incidents were resolved and closed in the system. The **closed_at** column was selected for creating the target feature. This is because it is more likely to accurate because tickets in IT incidents are only closed when some confirmation of the incident being resolved is received. The target variable was computed by subtracting the incident start time (the **open_at** field) from the incident closing time (the **closed_at** field). This calculation provided the target variable, representing the time taken to resolve the incident. The target variable was kept in days because hours and seconds were leading to large values which were not very suitable for learning. Certain fields, such as **caller_id** and **problem_id**, which were unique identifiers assigned to each sample, were dropped as they did not appear to have any predictive power regarding the target variable. Some variables such as category which are listed in the column named "Processed to continuous variable" in table 1 where text fields that contained numeric values which were relevant to predicting the target variable. A custom transformation was developed to extract relevant information from these columns. Details of the transformations and their functionality will be provided in the next section.

3.2 Methodology

Our goal was to implement and test the new approach for handling ordinal and categorical features. To accomplish this, custom transformers were designed using Scikit-learn (sklearn). A custom transformer named **cat2vec** was developed for handling nominal variables. A custom transformer named **cat2vec_wpos** was also developed to handle ordinal variables. The custom transformers were incorporated into Scikit-learn (sklearn) pipelines. These pipelines enable the saving of steps in a data processing operation, allowing for the execution of multiple experiments while rerunning the preprocessing steps in a defined format. Four pipelines were defined to test the performance of **cat2vec** and **cat2vec_wpos** and to compare their performance with existing popular techniques. This section provides the details of the custom transformers and pipelines used in this study.

3.2.1 Custom Transformers

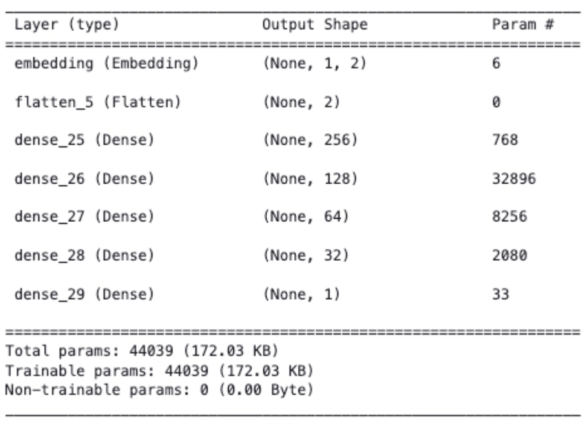
Three custom transformers were defined to facilitate the building of the experiments. The name and purpose of these transformers are mentioned in Table 2. Cat2vec and Cat2vec_wpos are the core of the new technique for handling nominal and ordinal variables.

Table 2. Customer Transformers

Transformer Name	Purpose
Cat2vec	To convert nominal features into dense vectors
Cat2vec_wpos	To convert ordinal features into dense vectors and to add positional encoding
Extract_Numeric	To extract numeric values from strings

3.2.1.1 Cat2vec

Cat2vec is custom transformer built in sklearn by inheriting from BaseEstimator and TranformerMixing classes. The constructor of cat2vec takes 1 input named nominal_variable which is a list containing the names of all nominal variables in the dataset. Cat2vec overrides the fit function which takes 2 inputs: dataset and target. Cat2vec then loops through each nominal variable. In each iteration it learns the vector embeddings of a nominal variable using a dense neural network containing 4 hidden layers. The input dimension for this neural network is number of unique values in the nominal variable (num_categories) whereas the output dimension is set to int (num_categories **0.5) +1. This formula is a heuristic that aims to provide a balance between capturing sufficient information from the categories and reducing the dimensionality to a manageable size. The square root of the number of categories (num_categories**0.5) is a common heuristic used in embedding layers to provide a compact yet informative representation. Adding 1 ensures that even if there is only one category, the embedding dimension is at least 1. A summary of the dense neural network used in cat2vec for num_categories = 3 is shown in Figure 1 below.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 1, 2)              6

 flatten_5 (Flatten)         (None, 2)                 0

 dense_25 (Dense)            (None, 256)               768

 dense_26 (Dense)            (None, 128)               32896

 dense_27 (Dense)            (None, 64)                8256

 dense_28 (Dense)            (None, 32)                2080

 dense_29 (Dense)            (None, 1)                 33

=================================================================
Total params: 44039 (172.03 KB)
Trainable params: 44039 (172.03 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig 1: Summary of neural network used by cat2vec

The architecture of this neural network is somewhat arbitrary and is built using the author’s experience with training neural networks. The fit() function of the neural network is called with inputs: nominal_variable and target. The neural network was trained for 15 epochs, which is an arbitrary number of epochs. It is acknowledged that further research is needed to identify a more optimal architecture and training regimen.

After learning the embeddings for a nominal variable, the embeddings are saved into a dictionary so that these embeddings can be used at the time of inference to map the nominal variables in the test set to their corresponding vector embeddings learnt during training. The code for cat2vec is available on GitHub in author’s repository here: https://github.com/adity-narvekar/cat2vec_with_position

3.2.1.2 Cat2vec_wpos

Cat2vec_wpos is also custom transformer which uses the same algorithm defined in the previous section. The only difference is that after learning the embeddings cat2vec_wpos then calls a function named getPositionEncoding function to get positional encodings for the ordinal variables. Positional encoding describes the location or position of an entity in a sequence so that each position is assigned a unique representation. Suppose if there is an ordinal variable with K unique values. The position encoding of the value at position “m” can be obtained using sine and cosine functions of varying frequencies. The formulas are as follows:

$$P(m, 2i) = sin\,(\frac{m}{n^{\frac{2i}{d}}})$$

$$P(m, 2i + 1) = \cos\,(\frac{m}{n^{\frac{2i}{d}}})$$

where:
m = Position of value in all values of ordinal variable.
d = dimension of the output positional embedding.
P (m, j) = Position function for mapping a position m in input to index (m, j) of the positional matrix.
N = user-defined scalar set to 10,000 by authors who proposed use of position encoding for NLP [4]
I = Value mapped to column indices.

The code for getPositionEncoding function can be seen in the cat2vec_wpos class available in author’s GitHub repo. https://github.com/adity-narvekar/cat2vec_with_position

3.2.1.3 Extract_numeric
Extract numeric is a simple class built to extract numeric values from text. The numeric values are then saved in separate columns and used as continuous features.

3.2.2 Pipelines
The custom transformers described in the previous section along with built-in transformers such as StandardScaler() are then packaged into pipelines for reproducibility. This study builds and uses 4 pipelines which are summarized in table 3.

Table 3: Pipelines built.

	Pipeline 1	**Pipeline 2**	**Pipeline 3**	**Pipeline 4**
Ordinal features	Ordinal Encoder + Standard Scaler	Ordinal Encoder + Standard Scaler	Cat2vec_wpos + Standard scaler	Cat2vec_wpos + Standard scaler
Nominal features	OneHot Encoder	Cat2vec + Standard scaler	OneHot Encoder	Cat2vec + Standard scaler
Continuous features	Standard Scaler	Standard Scaler	Standard Scaler	Standard Scaler

Each cell in this table mentions the transformations applied to the feature type represented by the row*

As seen in table 3, pipeline 1 applies ordinal encoder to ordinal variables and One Hot encoder to nominal variables. Pipeline 1 therefore represents the most common approach taken by machine learning practitioners. Pipeline 2 applies cat2vec to nominal variables while ordinal variables are processed using ordinal encoder. Pipeline 3 applies cat2vec_wpos to ordinal variables and one hot encoding to nominal variables and finally pipeline 4 applies the cat2vec_wpos to ordinal variables and cat2vec to nominal variables. Pipeline 4 therefore presents the new approach to handling nominal and ordinal features that we are presenting in this paper. A pictorial representation of all 4 pipelines is shown in Figure 2 for more clarity. The code for all pipelines is available in author's GitHub repo here: https://github.com/adity-narvekar/cat2vec_with_position.

Finally, each of these pipelines is used to transform the data, after which the transformed data is fitted into five popular machine learning algorithms. These algorithms are:

- Linear Regression
- K-nearest neighbors
- Random forest
- Support vector machines
- XGBoost

These algorithms were selected to encompass the major types of machine learning algorithms currently in use.

4. RESULTS
The experiments were conducted in AWS Sage Maker Studio, utilizing a machine with 32 GB of RAM and a single GPU for training. The code was written using standard open-source libraries. This study used sklearn for building the pipelines and Tensorflow for building the neural network used to learn the embeddings. The dataset was split with a 70-30 ratio for the training and test sets. The training set contained 99,198 samples, while the test set contained 42,514 samples. Each training run lasted approximately 1 hour. Running inference on test set took about 10-15 minutes for all 5 algorithms. This is because the test set had to be mapped to the embeddings learnt by cat2vec and cat2vec_wpos during training.

Figure 2 shows all the pipelines used to produce the results. Pipeline 1 represents standard and most widely used approach without any learned embeddings. Pipeline 2 represents the approach where only the embeddings for nominal variables are learnt using cat2vec while Pipeline 3 represents the approach where only the embeddings for ordinal variables are learnt using cat2vec_wpos. Finally, Pipeline 4 presents the new approach proposed by this study where embeddings are learnt for nominal and ordinal variables using cat2vec and cat2vec_wpos respectively.

The results of the experiments conducted using these Pipelines are presented in Table 4 and Table 5.

As seen in Table 4 the new approach proposed by this study called "cat2vec with position encoding" which is represented by pipeline 4 outperforms the approached represented by Pipelines 1-3. Also, as shown in Table 5, Pipeline 4 has the lowest MSE for test set for Random Forest, XGBoost and logistic regression.

Pipeline 4 when used with XGBoost produced the lowest MSE over train and test sets in all the experiments done in this study. The results indicate that the new approach proposed by this study outperforms the traditional approach (Pipeline 1) and the hybrid approaches (Pipeline 2 and 3).

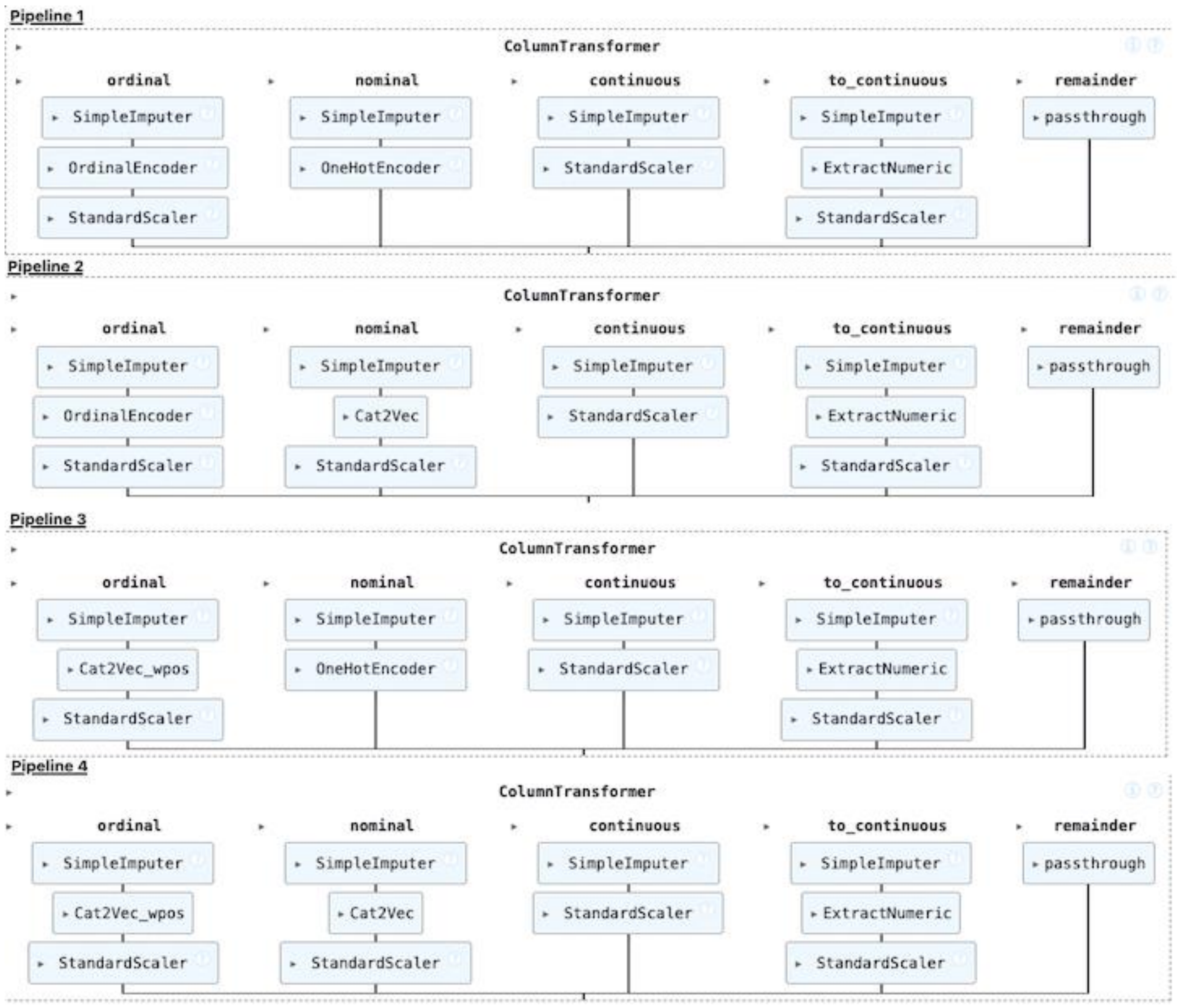


Fig 2: Pipelines built. Pipeline 4 represents the new approach proposed by this study.

Table 4: Lowest MSE for models trained using data transformed by Pipeline 1-4

	Incident Dataset							
	Pipeline 1		**Pipeline 2**		**Pipeline 3**		**Pipeline 4**	
Model	Train MSE	Test MSE	Train MSE	Test MSE	Train MSE	Test MSE	Train MSE	Test MSE
Random Forest	249.852	288.137	249.811	288.092	241.354	282.103	241.249	282.0644
SVM	404.015	421.22	412.238	429.916	393.485	409.692	400.746	417.657
XGBoost	134.174	188.7	136.16	190.582	138.124	188.39	132.095	182.595
Logistic Regression	337.04	354.364	337.026	354.345	329.76	346.187	329.757	346.11
KNN	166.342	270.993	167.169	273.398	172.014	281.402	173.128	283.472

Table 5: Pipelines with lowest test MSE for every algorithm used.

Model	Pipeline with lowest test MSE	Lowest Test MSE
Random Forest	Pipeline 4	282.0644
SVM	**Pipeline 3**	409.692
XGBoost	**Pipeline 3**	182.595
Logistic Regression	**Pipeline 3**	346.11
KNN	Pipeline 1	270.993

5. CONCLUSION

The new technique, named “Cat2vec with position encoding,” represented by Pipeline 4, outperformed the traditional method (Pipeline 1) and the hybrid methods (Pipelines 2 and 3) in the experiments conducted. “Cat2vec with position encoding” represented by Pipeline 4 combined with Random Forest, XGBoost and Logistic regression produced the lowest MSE with test sets as compared to other pipelines (see table 5). Also, “Cat2vec with position” when combined with XGBoost produces the lowest error on both train and test sets (see table 4). The results indicate a promising new approach that replaces nominal features with dense vectors and incorporates position encoding into dense vectors for ordinal features. These vectors are learnt during training using a deep neural network. For ordinal features there is an additional step that adds position encoding using a position encoding algorithm first used with transformers in natural language processing [4].

While the new technique named “Cat2Vec with position encoding” yields promising results, several areas for improvement are acknowledged and warrant further research. These considerations will be presented in the next section.

5.1 Future research

While a promising new technique is showcased in this paper, several areas for improvement are identified. Some of these ideas are listed below.

5.1.1 Neural network for learning embeddings

The neural network utilized for learning embeddings is the same for both **cat2vec** and **cat2vec_wpos**. This network is described in detail in Section 3.2.1.1. This neural network has a fixed architecture with 4 hidden layers. The number of hidden units in each layer have also been fixed for this study. This is an area where there is further scope of research where the neural network architecture should change based on the dataset or on the variable it is mapping to a new vector space.

5.1.2 Position encoding

The position encodings are learned using an algorithm initially introduced in natural language processing. This position encoding algorithm is explained in section 3.2.1.2. We have used the same hyper-parameters used by the authors of the position encoding algorithm [4]. This algorithm requires the length of output embeddings ex-ante. There is scope for researching this aspect to uncover heuristics for setting length of output embeddings based on number of unique values in an ordinal variable. Also, “n” described in section 3.2.1.2 is an input parameter which controls the positional encodings. This study has set “n” to 10,000 based on the original paper [4]. This leaves scope for tuning “n” further.

5.1.3 Use of different scalers

This study used only standard scalers in the pipelines shown in Figure 2. An interesting avenue for research would be to construct pipelines with different scalars such as min-max scaler, robust scaler etc. to check if further improvements are possible.

6. REFERENCES

[1] Huang, J., Li, Y.-F. and Xie, M. 2015. An empirical analysis of data preprocessing for machine learning-based software cost estimation. Information and Software Technology. 67, (Nov. 2015), 108–127. DOI: https://doi.org/10.1016/j.infsof.2015.07.004.Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[2] Huang, J., Li, Y.-F. and Xie, M. 2015. An empirical analysis of data preprocessing for machine learning-based software cost estimation. *Information and Software Technology*. 67, (Nov. 2015), 108–127. DOI: https://doi.org/10.1016/j.infsof.2015.07.004.

[3] Obaid, Hadeel & Ahmed Dheyab, Saad & Al-azzawi, Sana. (2019). The Impact of Data Pre-Processing Techniques and Dimensionality Reduction on the Accuracy of Machine Learning. 10.1109/IEMECONX.2019.8877011.

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I. 2017. Attention is All you Need. *arXiv (Cornell University)*. 30, (Jun. 2017), 5998–6008.

[5] Pargent, F., Pfisterer, F., Thomas, J. and Bischl, B. 2022. Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. *Computational Statistics*. 37, 5 (Mar. 2022), 2671–2692. DOI: https://doi.org/10.1007/s00180-022-01207-6.

[6] Pargent, F. 2019. A benchmark experiment on how to encode categorical features in predictive modeling. *https://osf.io/6fstx/*. (Mar. 2019).

[7] Potdar, K., S, T. and D, C. 2017. A comparative study of categorical variable encoding techniques for neural network classifiers. *International Journal of Computer Applications*. 175, 4 (Oct. 2017), 7–9. DOI: https://doi.org/10.5120/ijca2017915495.

[8] Cerda, P. and Varoquaux, G. 2022. Encoding High-Cardinality String categorical variables. *IEEE Transactions on Knowledge and Data Engineering*. 34, 3 (Mar. 2022), 1164–1176. DOI: https://doi.org/10.1109/tkde.2020.2992529.

[9] Golinko, E. and Zhu, X. 2018. Generalized feature embedding for supervised, unsupervised, and online learning tasks. *Information Systems Frontiers*. 21, 1 (Apr. 2018), 125–142. DOI: https://doi.org/10.1007/s10796-018-9850-y.

[10] Hancock, J.T. and Khoshgoftaar, T.M. 2020. Survey on categorical data for neural networks. *Journal of Big Data*. 7, 1 (Apr. 2020). DOI: https://doi.org/10.1186/s40537-020-00305-w.

[11] Maharana, K., Mondal, S. and Nemade, B. 2022. A review: Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*. 3, 1 (Jun.

2022), 91–99. DOI: https://doi.org/10.1016/j.gltp.2022.04.020.

[12] Gupta, S., Namdev, U., Gupta, V., Chheda, V. and Bhowmick, K. 2021. Data-driven preprocessing techniques for early diagnosis of diabetes, heart and liver diseases. *2021 Fourth International Conference on electrical, Computer and Communication Technologies (ICECCT)*. (Sep. 2021). DOI: https://doi.org/10.1109/icecct52121.2021.9616835

[13] Krishna, G.S., Supriya, K. and Rao, K.M. 2022. Selection of Data Preprocessing Techniques and Its Emergence Towards Machine Learning Algorithms using HPI Dataset. *2022 IEEE Global Conference on Computing, Power and Communication Technologies (GlobConPT)*. (Sep. 2022). DOI: https://doi.org/10.1109/globconpt57482.2022.9938255.

[14] Sukumar, P., Robert, L. and Yuvaraj, S. 2016. Review on Modern Data Preprocessing Techniques in Web Usage Mining (WUM). *IEEE*. (Oct. 2016). DOI: https://doi.org/10.1109/csitss.2016.7779441.

[15] Avanzi, B., Taylor, G., Wang, M. and Wong, B. 2024. Machine Learning with High-Cardinality Categorical Features in Actuarial Applications. *Astin Bulletin*. 54, 2 (Apr. 2024), 213–238. DOI: https://doi.org/10.1017/asb.2024.7.

[16] Kosaraju, N., Sankepally, S.R. and Rao, K.M. 2023. Categorical Data: need, encoding, selection of Encoding method and its Emergence in Machine Learning Models—A Practical Review Study on Heart Disease Prediction Dataset using Pearson Correlation. *Lecture notes in networks and systems*. 369–382.

[17] Destercke, S. and Yang, G. 2014. Cautious ordinal classification by binary decomposition. *Lecture notes in computer science*. 323–337.

[18] Bolikulov F, Nasimov R, Rashidov A, Akhmedov F, Cho Y-I. Effective Methods of Categorical Data Encoding for Artificial Intelligence Algorithms. *Mathematics*. 2024; 12(16):2553. https://doi.org/10.3390/math12162553

[19] Choong, A.C.H. and Lee, N.K. 2017. Evaluation of convolutionary neural networks modelling of DNA sequences using ordinal versus one-hot encoding method. *IEEE*. (Nov. 2017). DOI: https://doi.org/10.1109/iconda.2017.8270400.

[20] Yuan, Q., Chen, K., Yu, Y., Le, N.Q.K. and Chua, M.C.H. 2023. Prediction of anticancer peptides based on an ensemble model of deep learning and machine learning using ordinal positional encoding. *Briefings in Bioinformatics*. 24, 1 (Jan. 2023). DOI: https://doi.org/10.1093/bib/bbac630.

[21] K. Kunanbayev, I. Temirbek and A. Zollanvari, "Complex Encoding," *2021 International Joint Conference on Neural Networks (IJCNN)*, Shenzhen, China, 2021, pp. 1-6, DOI: 10.1109/IJCNN52387.2021.9534094.

[22] Dahouda, M.K. and Joe, I. 2021. A Deep-Learned embedding technique for categorical features encoding. *IEEE Access*. 9, (Jan. 2021), 114381–114391. DOI: https://doi.org/10.1109/access.2021.3104357.

[23] Chen, P.-C., Tsai, H., Bhojanapalli, S., Chung, H.W., Chang, Y.-W. and Ferng, C.-S. 2021. A simple and effective positional encoding for transformers. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. (Jan. 2021). DOI: https://doi.org/10.18653/v1/2021.emnlp-main.236.

[24] Ke, G., He, D. and Liu, T.-Y. 2021. Rethinking positional encoding in language pre-training. *International Conference on Learning Representations*. (May 2021).

[25] Gil Press (2021) Andrew Ng Launches a Campaign for Data-Centric AI. Forbes. Available from: https://www.forbes.com/sites/gilpress/2021/06/16/andrew-ng-launches-a-campaign-for-data-centric-ai/?sh=1b802f8d74f5.