IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A Deep-Learned Embedding Technique for Categorical Features Encoding

MWAMBA KASONGO DAHOUDA, INWHEE JOE
Department of Computer Science, Hanyang University, Seoul 04763, South Korea

Corresponding author: Inwhee Joe (iwjoe@hanyang.ac.kr)

This work was supported by the Institute for Information and Communication Technology promotion (IITP) grant funded by the Korea government (MSIP) (Development of the technology to automate the recommendations for big data analytic models that define data characteristics and problems), under Grant 2020-0-00107.

ABSTRACT Many machine learning algorithms and almost all deep learning architectures are incapable of processing plain texts in their raw form. This means that their input to the algorithms must be numerical in order to solve classification or regression problems. Hence, it is necessary to encode these categorical variables into numerical values using encoding techniques. Categorical features are common and often of high cardinality. One-hot encoding in such circumstances leads to very high dimensional vector representations, raising memory and computability concerns for machine learning models. This paper proposes a deep-learned embedding technique for categorical features encoding on categorical datasets. Our technique is a distributed representation for categorical features where each category is mapped to a distinct vector, and the properties of the vector are learned while training a neural network. First, we create a data vocabulary that includes only categorical data, and then we use word tokenization to make each categorical data a single word. After that, feature learning is introduced to map all of the categorical data from the vocabulary to word vectors. Three different datasets provided by the University of California Irvine (UCI) are used for training. The experimental results show that the proposed deep-learned embedding technique for categorical data provides a higher F1 score of 89% than 71% of one-hot encoding, in the case of the Long short-term memory (LSTM) model. Moreover, the deep-learned embedding technique uses less memory and generates fewer features than one-hot encoding.

INDEX TERMS Data Preprocessing, Categorical Variables, Natural Language Processing, Machine Learning.

I. INTRODUCTION

MANY machine learning algorithms require that their input is numerical; therefore, categorical features must be transformed into numerical features before fitting them into an algorithm [1]. Natural language processing (NLP) is a field of artificial intelligence that studies the interactions between computers and human languages, in particular how to program computers to process and analyze large amounts of natural language data. Many modern NLP systems and approaches regard words as atomic units, with no concept of word similarity since indices in a vocabulary are used to represent them [2]. Text classification is the problem of assigning categories to text data according to its content. Categorical data are commonplace in many data science and machine learning problems but are usually more challenging to deal with than numerical data. Preprocessing categorical variables becomes important since most machine learning models only consider numerical variables; therefore, we must transform these categorical variables to numbers in order for the model to comprehend and retrieve useful information. There are many ways to encode categorical variables for modeling, and one of the most commonly used encoding techniques [3] is one-hot encoding. This is where each level of the categorical variable is compared to a specified reference level, especially when there is no natural ordering between the categories, e.g., a feature 'City' with names of cities such as 'Seoul','Paris', 'Kinshasa'. Categorical features are prevalent and frequently have a high degree of cardinality. Some categorical encoding approaches have been studied in the statistical-learning field in [4]. In [5], the issue of encoding in the presence of errors and how to encode categories that are not present in the training set have been ignored. However, one-hot encoding produces extremely

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3104357, IEEE Access

high-dimensional vector representations in such situations, posing memory and computability issues for machine learning models. Furthermore, the word representations produced by one-hot encoding or hashing are sparse, high-dimensional, and hardcoded [6].

A good understanding of data is essential for accurate analysis. Before proceeding to the actual analysis, the data is processed to aid algorithms and to improve efficiency. There are other ways of classifying variables that are common in statistics as described in Fig. 1. On one side, we have the qualitative variables which are descriptive/categorical; many statistics such as mean and standard deviation, do not make sense to compute with qualitative variables, and on the other side we have the quantitative variables that have numeric meaning; therefore, statistics like means and standard deviations make sense.

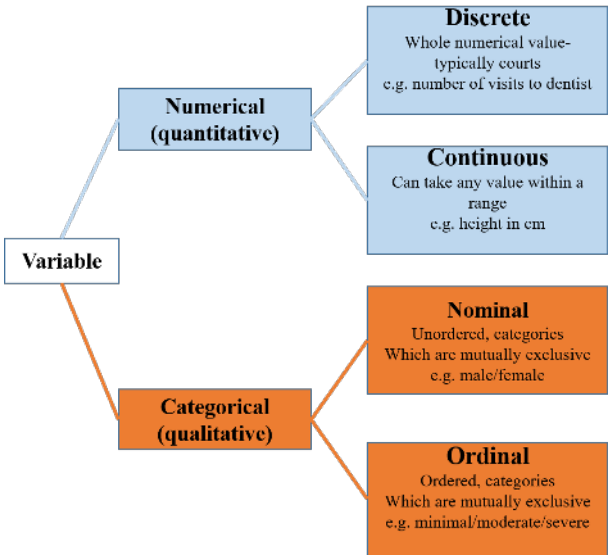

FIGURE 1. Variables: Quantitative (Numerical) vs Qualitative (Categorical).

Data variables generally fall into one of the four categories: nominal scale, ordinal scale, discrete, and continuous [7]. In this study we are going to focus on two of them, thus using the deep-learned embedding technique, we can easily encode nominal or ordinal variables.

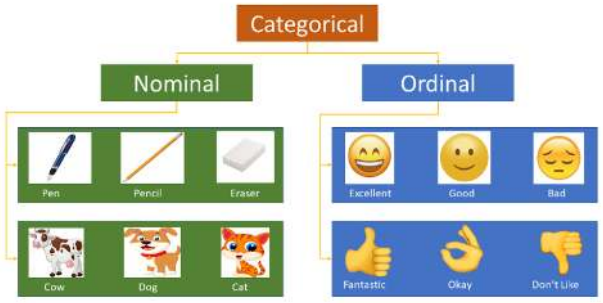

FIGURE 2. Categorical data: Nominal vs ordinal scale.

Categorical data can be divided into two groups, as described in Fig. 2, which are nominal (no particular order) and ordinal (with some particular order) [8].

Typical examples of nominal variables include genotype, blood type, zip code, gender, race, eye color.

A nominal scale describes a variable with categories that do not have a natural order or ranking. You can encode nominal variables with numbers if you want but the order is arbitrary and any calculations such as computing a mean, median, or standard deviation would be meaningless.

An ordinal scale is one where the order matters but not the difference between values. Typical examples of ordinal variables include satisfaction rating ("extremely dislike", "dislike", "neutral", "like", "extremely like") [9].

A. CATEGORICAL VARIABLE ENCODING TECHNIQUES

1) Label Encoding or Ordinal Encoding

We use this categorical data encoding technique when the categorical feature is ordinal. In this case, retaining the order is important. Hence, encoding should reflect the sequence. In Label encoding, each label is converted into an integer value. For example, create a variable that contains the categories representing the education qualification of a person [10].

2) One-hot Encoding

We use this categorical data encoding technique when the features are nominal (do not have any order). In one-hot encoding, for each level of a categorical feature, we create a new variable, and each category is mapped with a binary variable containing either 0 or 1. Here, 0 represents the absence, and 1 represents the presence of that category [10].

3) Dummy Encoding

Dummy coding scheme is similar to one-hot encoding. This categorical data encoding method transforms the categorical variable into a set of binary variables (also known as dummy variables). In the case of one-hot encoding, for N categories in a variable, it uses N binary variables.

4) Effect Encoding

This encoding technique is also known as Deviation Encoding or Sum Encoding. Effect encoding is almost similar to dummy encoding, with a little difference. In dummy coding, we use 0 and 1 to represent the data but in effect encoding, we use three values i.e. 1, 0, and -1.

5) Hash Encoder

To understand Hash encoding it is necessary to know about hashing. Hashing is the transformation of arbitrary size input in the form of a fixed-size value. We use hashing algorithms to perform hashing operations i.e to generate the hash value of an input.

6) Binary Encoding

Binary encoding is a combination of Hash encoding and one-hot encoding. In this encoding scheme, the categorical

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3104357, IEEE Access

feature is first converted into numerical using an ordinal encoder. Then the numbers are transformed in the binary number. After that binary value is split into different columns.

7) Base N Encoding

For binary encoding, the base is 2 which means it converts the numerical values of a category into its respective binary form. If you want to change the base of the encoding scheme you may use the base N encoder. In the case when categories are more and binary encoding is not able to handle the dimensionality then we can use a larger base such as 4 or 8.

8) Target Encoding

Target encoding is a Baysian encoding technique. In target encoding, we calculate the mean of the target variable for each category and replace the category variable with the mean value. In the case of the categorical target variables, the posterior probability of the target replaces each category.

B. ARTIFICIAL NEURAL NETWORKS

An artificial neural network (ANN) is the piece of a computing system designed to simulate the way the human brain analyzes and processes information. It is the foundation of artificial intelligence (AI) and solves problems that would prove impossible or difficult by human or statistical standards. ANNs have self-learning capabilities that enable them to produce better results as more data becomes available. The neural network gains the experience initially by training the system to correctly identify pre-selected examples of the problem. The response of the neural network is reviewed and the configuration of the system is refined until the neural networks analysis of the training data reaches a satisfactory level.

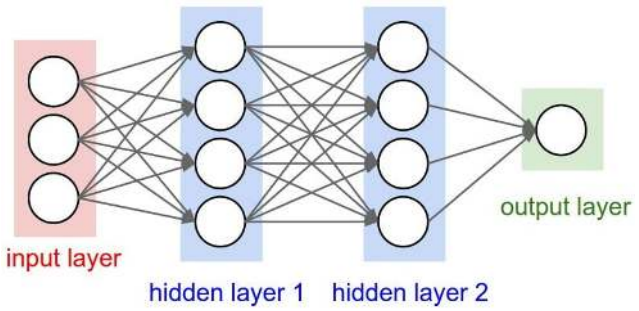

FIGURE 3. Basic artificial neural networks.

In addition to the initial training period, the neural network also gains experience over time as it conducts analyses on data related to the problem. Classification using ANN is one of the most dynamic research and application areas; and as shown in Fig. 3, ANN is widely used for classification purposes because of its ability to generalize and map input-output relations based on existing data [11]. Deep neural network architectures have recently gained a lot of traction in the NLP community [12] due to their ability to allow researchers to build and train deep neural networks, implement vectorized neural networks [13], and identify architecture parameters [14]. Moreover, it allows building recurrent neural networks (RNN) and its variants (Gated recurrent units: GRUs, Long short-term memory: LSTMs) in a variety of NLP applications [15] such as character-level language modeling, word segmentation [16], [17], and word embedding [18].

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed; it focuses on the development of computer programs that can change when exposed to new data. In both, regression and classification analysis, categorical variables are widely used; however, machine learning algorithms accept only numeric values as input. Whenever we want to use categorical data for machine learning purposes; the data needs to be encoded into numeric values such that each categorical feature is represented with a number [19]. Additionally, categorical data can be considered as a word; therefore, it can be embedded on the basis of word embedding techniques where each word in a particular language is allocated to a high-dimensional vector in word embedding models, with the geometry of the vectors capturing semantic relationships between the words [20]. Many researchers have investigated word embedding [21]–[23]; and the emergence of artificial neural networks in natural language processing is mostly based on word embedding [24], when compared to one-hot encoding, this method brings words with similar meanings closer together in word space, improving word continuity.

In this paper, a deep-learned embedding technique for categorical data encoding on a categorical dataset is presented. Our technique is based on word embedding which is also a part of a deep learning model. Here, we consider each categorical variable as a single word, or as a token so that the distributed word representations can be applied. Therefore, the idea is to represent words as feature vectors or word vectors; and then each entry in vector stands for one hidden feature inside the word meaning. Using the Keras model, we build our model with an embedding layer that can be used for neural networks on text data. We have used three different datasets from the University of California Irvine (UCI) for experimentation.

The results show that the data encoded with the embedding technique give the highest accuracy as compared to the other techniques and can support nominal and ordinal categorical variables, and also our technique is used where there is vector similarity or not. We compare our technique with one-hot encoding; and finally, our deep-learned embedding technique generated fewer features and used less memory than one-hot encoding. Our contributions can be summarized as follows:

- We propose a new technique, deep-learned embedding, to encode categorical variables that can be ordinal or

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3104357, IEEE Access

nominal. It encodes all words, regardless of how similar they are.

- We create a corpus by separating categorical data from numerical to precisely have the vocabulary, and then we embed all the categorical data to words vectors at a time, which makes the deep-learned embedding technique achieve high performance with low computation cost and less memory usage.

- In experiments, we demonstrate the effectiveness of our method by encoding the categorical data of three different datasets: bank marketing dataset (BMD), adult income dataset (AID), and in-vehicle coupon recommendation dataset (I-VCRD); and building different machine learning models. Importantly, for the long-short term memory model, it achieves a better accuracy of 64% and a better F1 score of 0.86 compared to when we use one-hot encoding on the in-vehicle coupon recommendation dataset.

- Furthermore, the deep-learned embedding technique uses less memory and also generates fewer features than one-hot encoding. For instance, when using the in-vehicle coupon recommendation dataset, our method generates 80 features and uses 6.494 MB compare to the one-hot encoding that generates 100 features and uses 10.174 MB.

The paper is organized as follows: Section 2 presents related work, and section 3 describes our proposed method, and the following section 4 the experimental results. In section 5, we present the limitations of the study, and then we give our conclusion in the last section.

II. RELATED WORK

Most of the literature on encoding categorical variables relies on the idea that the set of categories is finite, known a priori, and composed of mutually exclusive elements [25]. In [26], the author presents a comparative study of categorical variable encoding techniques for neural network classifiers by covering seven techniques for encoding categorical variables; on the other hand, the authors evaluate each technique on the UCI Cars dataset with one neural network architecture. Beyond one-hot encoding, the statistical-learning literature has considered other categorical encoding methods in previous work [27]–[29]. In [30], Data representation learning was presented, and this approach was used to evaluate both classic feature learning methods and state-of-the-art deep learning models. One-hot encoding is the most widely used coding scheme; it compares each level of the categorical variable to a fixed reference level, and also it transforms a single variable with n observations and d distinct values to d binary variables with n observations each. Each observation indicating the presence (1) or absence (0) of the dichotomous binary variable [31]. With ordinal encoding, an integer is assigned to each category; therefore, the provided number of existing categories is known. It does not add any new columns to the data; however, it implies an order to the variable that may not actually exist [32]. Compare to all the above categorical variable encodings, our technique is based on word embeddings where each categorical variable will be represented as a multidimensional array or feature vector or word vector. A table of a comparison of random forest accuracy for various encoding techniques shown that the one-hot encoding has a higher dimensionality than other encoding schemes [33]. Word embedding has also been considered as entity embedding in [34], this research has shown that a neural network can learn the mapping during a typical supervised training phase; therefore, with the development of entity embeddings, there has been a recent advance in categorical variable representation [35]–[38]. Moreover, when compared to the frequently used one-hot encoding, the introduction of word embeddings not only lowered memory use but also enhanced the machine learning algorithms learning ability from data [39]. In our work, we demonstrated that encoding categorical variables based on word embedding use not only less memory but also generates fewer features.

III. PROPOSED METHOD

It is more important to know what coding scheme we should use having into consideration the dataset we are working on, and the model we are going to use. The current research was developed in three stages: (1) preprocess the data, (2) build the neural network, (3) and finally, analyze the metrics and compare them between different encoding techniques.

1) Preprocessing the data

The first stage is very important because preprocessing the data is one of the major steps when we are dealing with any kind of text model. This step never has one hot rule, and totally depends on the problem statement. During this stage, we have to look at the distribution of our data, what techniques are needed and how deep we should clean. We first start by splitting the data into two subsets, numerical data and categorical data that allows us to create a corpus that contains only categorical data, and then we convert to lowercase all the data and our label into a numeric form. Thus, according to the description of datasets, all three datasets are used for classification problems; therefore, their outcome is a binary output (yes: 1 or no: 0). We say if a client can subscribe for a deposit then set it equal to one otherwise, set it equal to zero, using for instance the bank marketing dataset. The bag-of-words model is simple, it builds a vocabulary from a corpus of documents and counts how many times the words appear in each document; we use the word2vec in order to produce a vector space, typically of several hundred dimensions, with each unique categorical variable in the corpus such that variable that shares common contexts in the corpus are located close to one another in the space. Therefore, we used the term

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3104357, IEEE Access

frequency-inverse document frequency (TFIDF) as score for term i in document j as shown in Equation (1), in order to discover how many times a categorical variable appears in our corpus.

$$TFIDF = TF(i,j) * IDF(i) \tag{1}$$

where
IDF = Inverse Document Frequency
TF = Term Frequency
i = Term or categorical variable
j = Document or corpus or a subset that contains only categorical variable.

$$TF(i,j) = \frac{Term\ i\ frequency\ in\ document j}{Total\ words\ in\ document} \tag{2}$$

$$IDF(i) = \log_2(\frac{Total\ documents}{document\ with\ term\ i}) \tag{3}$$

The above Equation (2) and (3) can be generalized into one component. The actual mathematical formula become :

$$w_{i,j} = tf_{i,j} * \log(\frac{N}{df_i}) \tag{4}$$

$\text{tf}_{i,j} = number\ of\ occurrences\ of\ i\ in\ j$
$df_i = number\ of\ documents\ containing\ i$
$N = total\ number\ of\ documents$

As shown in Fig. 4, we used the vector space models to discover relationships between categorical variables and visualize those relationships in the corpus. The figure below shows us the categorical data into in two-dimensions space (2D space) where we can see categorical data such as "feb", "oct", "jan", "yes", "no", "nov", "retired". As we mentioned above, on Fig. 4, we can see that variable that shares common contexts in the corpus are located close to one another in the space.

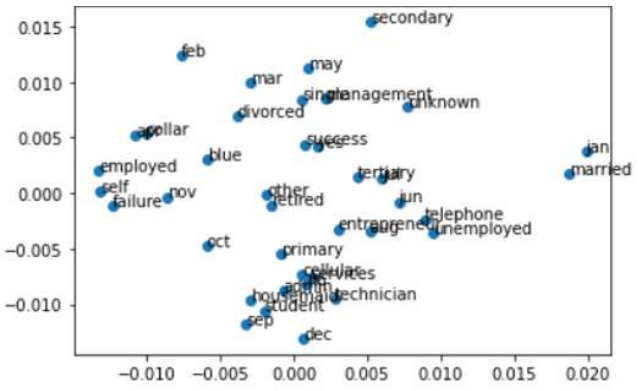

FIGURE 4. Categorical data from Bank marketing dataset embedded in 2D vector space.

There is some similarity between words. In natural language processing, useless words such as "the", "me", "yes" and "no", are referred to as stop words. Therefore, it is a common practice to remove the stopwords while preprocessing the text data because they take up space in the dataset and take up valuable processing time that could have an impact while training the model.

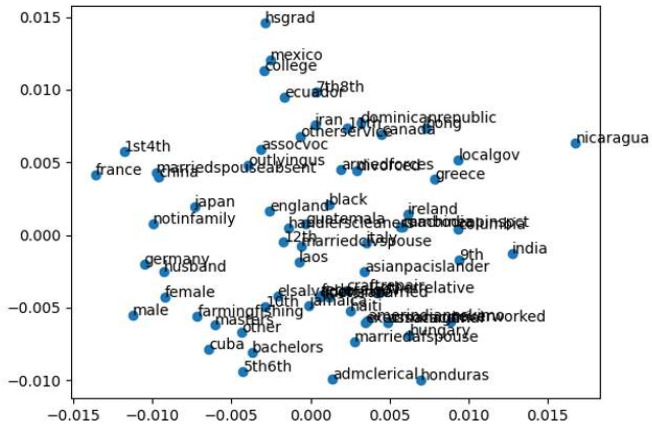

FIGURE 5. Categorical data from Adult Income dataset embedded in 2D vector space.

Figure 5 depicts categorical data from the adult income dataset in two-dimensions space; similarly to Fig. 4, we can observe that variables with similar contexts in the corpus are clustered together in the space. Categorical data from the In-

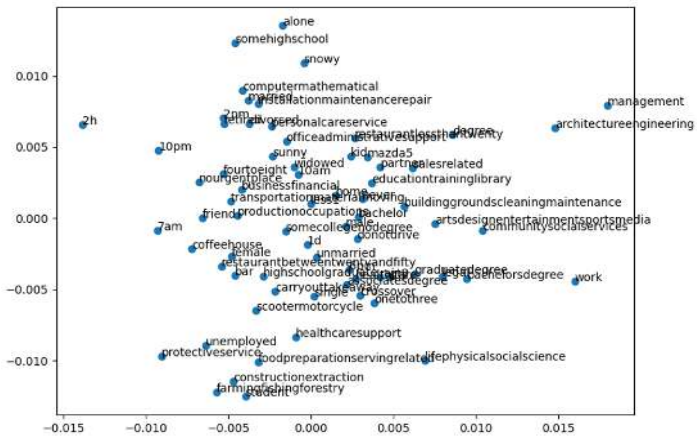

FIGURE 6. Categorical data from In-vehicle coupon recommendation dataset embedded in 2D vector space.

vehicle coupon recommendation dataset can also be seen in two-dimensions space as shown in Fig. 6. Importantly, we have to consider each categorical variable as a single word in order to tokenize correctly the categorical data.

2) Deep-learned embedding technique

In this part, we are going to describe how our technique works. In natural language processing (NLP), tokenization plays a significant role in dealing with text data; therefore, it is a way of separating a piece of text into smaller units called tokens. Here, tokens can be either word, characters, or subwords. Hence, tokenization can be broadly

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3104357, IEEE Access

classified into 3 types: word, character, and subword (n-gram characters) tokenization. Representing strings as n-grams at the character level is similar to vectorizing text as tokens or words [40]; therefore, in our case, we have used the word tokenization (1-gram sequence or unigram) in order to have each categorical data as a single word. First, we create the tokenizer object, providing the maximum number of data to keep in our vocabulary, after the tokenization process, every categorical data will be considered as a token in our corpus as shown in Fig. 7. Considering the bank marketing dataset, this dataset contains many "yes" and "no", and we cannot remove them because they represent useful information in terms of making a decision whether the client can subscribe for a deposit or not based on the description of the dataset.

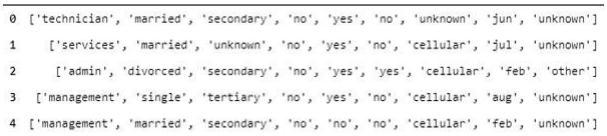

FIGURE 7. Categorical data tokenized: Bank marketing dataset.

During the preprocessing stage, all categorical data have been tokenized in the created corpus. As shown in Fig. 7, we did not remove stopwords because they contain important information for this specific classification problem. The stopwords are considered as noise for the model; consequently, they can slow down the training process that could also give a lower accuracy. Therefore, to cope with this problem, we have built different machine learning models with hyperparameter optimization tuning in order to have higher accuracy and a good model.

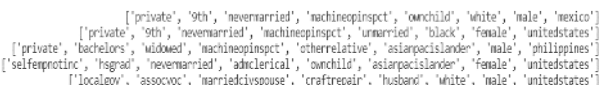

FIGURE 8. Categorical data tokenized: Adult income dataset.

The categorical data from the Adult income and In-vehicle coupon recommendation datasets have also been tokenized as illustrated in Fig. 8 and 9.

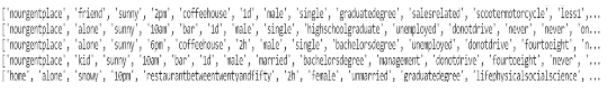

FIGURE 9. Categorical data tokenized: In-vehicle coupon recommendation dataset.

After tokenizing our data, we split the subset into training and testing data. Accordingly, 80% were allocated to the training set and 20 % were allocated to the test set, and we saved the training and test sets into a CSV file to ensure we are using the same data for each machine learning model. Now We have the vocabulary of all the data, the training and testing set, and we apply a vectorizer to the training and testing set separately; therefore, all the tokenized data can be learned from the vocabulary where tokens are terms and values are indices in the feature matrix; resultantly, we have the TF-IDF-weighted document-term matrix or the feature matrix which is a sparse matrix. This process can also be called feature learning techniques where categorical data tokenized from the vocabulary are mapped to word vectors of real numbers. We have finished encoding our categorical data into embedding matrix; before further process; firstly, we have to convert the sparse matrix into a dataframe; secondly, we reshape the numerical subset, and finally concatenate the encoded categorical data with the numerical data subset.

3) Model description

We use different machine learning models such as Logistic Regression (LR), Multi-Layer Perceptron (MLP), Random Forest (RF), Gradient Boosting (GB), and Deep Learning Model (DL), and we train and evaluate these models using the training and testing data encoded with deep-learned embedding technique.

It is a good idea to build a deep learning model with the Keras model because embeddings can be used in Keras via the embedding layer. Therefore, we built powerful recurrent neural architectures [41] using deep neural networks; and we have used the sequential Keras model to build neural networks for classification tasks on three different datasets. We use the embedding matrix in the first embedding layer of the neural network. Each id in the input sequence will be used as the index to access the embedding matrix. Our neural network model is a three built-in recurrent neural network (RNN) layers in Keras; this sequential model processes sequences of integers, embeds each integer into a dimensional vector, then processes the sequence of vectors using a LSTM layer.

We apply a grid search by setting up a model generator function, setting up a parameter grid and doing a grid search with cross-validation, and after the training, the outcomes of our grid search can be reported where we can see different models with its parameters; therefore, we can pick the best model. The scikit-learn have been used, a toolkit often used with Keras and other machine learning software in order to perform the grid search and obtain the classification report, which will tell us about our best model. Then, we have to import Keras's KerasClassifier wrapper, which makes it compatible with scikit-learn. We apply grid search for deep learning models for hyperparameter optimization so that the model can grid search the epochs, the activation function, and the optimizer; hence, it will give the best model along with its parameters. Now, we set up our KerasClassifier, handing it the model-building function, set verbose to 0 to hide the progress bars of each Keras run, we also set up a grid search with cross-validation. For its estimator, we give it our model, which is our KerasClassifier wrapper, and our grid parameter; then we say cv=5, meaning cut the data (the training data) into five different segments and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3104357, IEEE Access

then cross-validate. Train on 4, and use one fifth to validate and iteratively repeat this in order to search for the best hyperparameter values. we call the fit function going from our x training data (again, those are our input training data) to our y training data (these are the labels from the label encoded) and then print out our best results.

IV. EXPERIMENTAL RESULTS

A. DATASET DESCRIPTION

1) The Bank Marketing Dataset

The Bank Marketing dataset[1] used for this study was obtained from the UCI Machine Learning Repository. The data is related to direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe to a term deposit (variable y). The data includes information of 45211 clients. Each record has 17 attributes. This dataset contains 10 columns of categorical data and 7 columns of numerical data.

2) Adult Income Dataset

The Adult Income Dataset[2] contains 48842 samples of income data from the census bureau database for individuals in the United States. Also known as "Census Income" dataset. The aim is to determine whether a person's salary exceeds $50,000 USD. The data includes information of 48842 person. Each record has 14 attributes composed of 8 columns of categorical data and 6 columns of numerical data.

3) In-vehicle Coupon Recommendation Dataset

The In-vehicle Coupon dataset[3] contains 12684 samples of coupon recommended. This data studies whether a person will accept the coupon recommended to him in different driving scenarios. Each record has 26 attributes composed of 16 columns of categorical data and 10 columns of numerical data.

B. PERFORMANCE RESULTS

Classification accuracy of different models was measured, and the results from the experiments performed on the Bank Marketing Dataset are illustrated in the different tables below. Using a one-hot and ordinal encoding technique for the same dataset shows low accuracy compared to our deep-learned embedding technique. The deep-learned embedding technique gives better accuracy of 73% than one-hot encoding which gives 71% of accuracy as shown in table 3. Besides the Bank Marketing dataset, we also use 2 more categorical datasets: Adult Income and In-vehicle coupon datasets. In view of the In-vehicle coupon recommendation dataset, our approach generates 80 features and uses 6.494 MB compared to 100 features and 10.174 MB for one-hot encoding as shown in Table 7. The use of embeddings reduced the memory usage [39] compared with commonly used one-hot encoding. More importantly, evaluating a machine learning algorithm is an essential part of any deep learning project; since our model may give us satisfying results when evaluated using a metric like accuracy but may give poor results when evaluated against other metrics such as mean square error or any other metric. Most of the time we use classification accuracy to measure the performance of our model; however, it is not enough to truly judge our model. Therefore, we evaluate the performance of different machine learning models by using the following metrics:

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ numbers\ of\ predictions\ made} \quad (5)$$

Accuracy: It is the ratio of number of correct predictions to the total number of input samples.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (6)$$

Precision : It is the number of correct positive results divided by the number of positive results predicted by the classifier.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (7)$$

Recall: It is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). There is also another metric that we can use if we want to seek a balance between Precision and Recall which F1 Score. It is also called the F-score or the F-measure. It might be a better measure to use if we need to convey a balance between Precision and Recall, and it is calculated as follows:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (8)$$

Area Under Curve (AUC) is one of the most widely used metrics for evaluation. It is used for binary classification problems. The area under the curve (AUC) indicates the probability that the classifier will rank a randomly chosen positive observation higher than a randomly chosen negative one.

$$Mean\ Square\ Error = \frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_i)^2 \quad (9)$$

Mean Squared Error (MSE) is quite similar to Mean Absolute Error, the only difference being that MSE takes the average of the square of the difference between the original values and the predicted values.

There are many ways to encode categorical variables as numbers and fit them into an algorithm. Therefore, we have used different metrics together with different encoding techniques including the deep-learned embedding techniques in order to evaluate different machine learning algorithms. The results presented in Table 1 show that when

[1] https://archive.ics.uci.edu/ml/datasets/Bank+Marketing
[2] https://archive.ics.uci.edu/ml/datasets/adult
[3] https://archive.ics.uci.edu/ml/datasets/in-vehicle+coupon+recommendation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3104357, IEEE Access

encoding categorical data with target encoding, the gradient boosting outperforms the other models and give higher accuracy along with the precision. Using binary encoding,

TABLE 1. Comparison between Target and Binary Encoding

Bank Marketing dataset				
Encoding	Model	Accuracy	Precision	Recall
Target	LR	0.69	0.71	0.63
	MLP	0.69	0.71	0.62
	RF	0.69	0.76	0.53
	GB	0.71	0.78	0.52
Binary	LR	0.62	0.61	0.63
	MLP	0.68	0.70	0.56
	RF	0.70	0.73	0.58
	GB	0.70	0.75	0.53

we also built different machine learning models, and the result shows that when compared to target encoding, binary encoding gives a low accuracy and low precision as shown in Table 1. The experiment results, as shown in Table 2

TABLE 2. Comparison between Target and Binary Encoding

Bank Marketing dataset			
Encoding	Model	AUC	MSE
Target	Logistic Regression	0.694	0.302
	Multi-Layer Perceptron	0.696	0.300
	Random Forest	0.689	0.303
	Gradient Boosting	0.692	0.299
Binary	Logistic Regression	0.621	0.371
	Multi-Layer Perceptro	0.678	0.323
	Random Forest	0.694	0.305
	Gradient Boosting	0.682	0.300

show that values of the AUC and MSE are almost the same for both target and binary encoding. One-hot encoding is the most used encoding technique in many projects when it comes to encode categorical variables; this technique maps each category to a vector that contains 1 and 0 denoting the presence or absence of the feature. The number of vectors depends on the number of categorical features. This method produces a lot of columns that slow down the learning significantly if the number of the category is very high [33], and it can also have a higher dimensionality than other encoding schemes.

For the reason that the one-hot is the most used encoding technique for categorical data; we made a deep comparison between one-hot encoding and our deep-learned embedding technique. Our proposed technique, deep-learned embedding technique, gives higher accuracy than one-hot encoding, and further the precision is 0.83 higher than 0.79 for one-hot encoding as shown in Table 3.

Besides the Bank marketing dataset, we also applied our technique on the Adult income dataset, and compared the result to one-hot encoding. Repeatedly, encoding categorical variable with our proposed technique and fit them into machine learning model gives almost similar result as when using one-hot encoding. Here, considering the accuracy, as

TABLE 3. Comparison between One-hot and Deep-learned embedding

Bank Marketing dataset				
Encoding	Model	Accuracy	Precision	Recall
One-hot	LR	0.69	0.75	0.55
	MLP	0.71	0.79	0.56
	RF	0.70	0.75	0.58
	GB	0.71	0.78	0.56
	LSTM	0.71	0.79	0.56
Our technique	LR	0.65	0.62	0.52
	MLP	0.68	0.73	0.59
	RF	0.70	0.74	0.58
	GB	0.70	0.76	0.57
	LSTM	0.73	0.83	0.57

TABLE 4. Comparison between One-hot and Deep-learned embedding

Adult Income dataset				
Encoding	Model	Accuracy	F1 Score	MSE
One-hot	LR	0.78	0.76	0.19
	MLP	0.77	0.70	0.22
	RF	0.79	0.86	0.133
	GB	0.78	0.85	0.135
	LSTM	0.77	0.71	0.126
Our technique	LR	0.76	0.86	0.241
	MLP	0.76	0.71	0.435
	RF	0.76	0.65	0.24
	GB	0.75	0.86	0.54
	LSTM	0.76	0.89	0.18

shown in Table 4, one-hot encoding surpasses our proposed technique; however, our proposed technique outperforms a higher F1 score of 89% than 71% of one-hot encoding, in consideration of the LSTM model.

The results shown in Table 5 show again that using our proposed technique to encode categorical variables can help the machine learning models to have good accuracy and better F1 score than when using one-hot encoding. For instance, LSTM outperforms 65% of accuracy compared to 61% of one-hot encoding, and further, the F1 score metric is 0.86 higher than 0.66 for one-hot encoding as shown in Table 5.

TABLE 5. Comparison between One-hot and Deep-learned embedding

In-vehicle coupon recommendation dataset				
Encoding	Model	Accuracy	F1 Score	MSE
One-hot	LR	0.56	0.48	0.43
	MLP	0.57	0.43	0.43
	RF	0.69	0.73	0.25
	GB	0.66	0.69	0.29
	LSTM	0.61	0.66	0.23
Our technique	LR	0.60	0.73	0.40
	MLP	0.61	0.73	0.41
	RF	0.62	0.61	0.37
	GB	0.62	0.60	0.37
	LSTM	0.64	0.86	0.24

We have used different models and different evaluations to ensure that our deep-learned embedding technique is preferable when working on categorical dataset. As shown

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3104357, IEEE Access

TABLE 6. Comparison between One-hot and Deep learned embedding

Bank Marketing dataset			
Encoding	Model	AUC	MSE
One-hot	Logistic Regression	0.695	0.300
	Multi-Layer Perceptron	0.711	0.282
	Random Forest	0.700	0.293
	Gradient Boosting	0.684	0.298
	LSTM	0.682	0.300
Our technique	Logistic Regression	0.614	0.301
	Multi-Layer Perceptron	0.662	0.310
	Random Forest	0.695	0.300
	Gradient Boosting	0.694	0.301
	LSTM	0.697	0.293

Table 6, when using one-hot encoding, logistic regression, multi-layer perceptron, and random forest give a higher value of AUC than our technique. On the other hand, our deep-learned embedding technique surpasses the one-hot when using gradient boosting and the LSTM model. The AUC is used for binary classification and this is better for our study because taking into consideration the bank marketing dataset; we are predicting whether a client can subscribe for a deposit or not which means it is a binary classification problem. Moreover, our proposed technique works well with artificial neural networks such as LSTM because it is based on word embedding; as shown in Fig. 10, deep-learned has better accuracy than one-hot encoding.

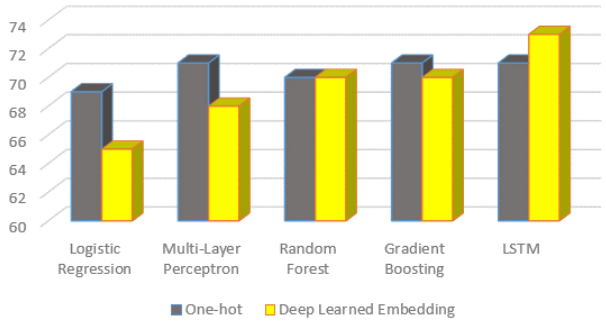

FIGURE 10. Comparison of accuracy between One-hot and Deep-learned embedding

Another important point while dealing with categorical variable encoding is the number of features that can be generated after encoding; usually, after encoding we obtain a sparse matrix that contains many zero. Besides the number of generated features after encoding, it is also essential to know the memory used by the dataframe. In this way, on one side we have the number of the generated features, and on the other side we have the capacity memory used by the data; getting to know how much memory that has been used by a data frame can be extremely useful when working with a bigger data frame. Therefore, in order to check the memory usage (in Megabytes: MB), we first convert the sparse matrix into a data frame and then calculate the memory usage. Here, note that before preprocessing data as shown in Table 8, our data frame had 7.2 MB, 3.91 MB, and 2.05 MB for the Bank marketing dataset, Adult income dataset, and In-vehicle coupon dataset respectively.

TABLE 7. Comparison of Generated features between One-hot and Deep-learned embedding

Bank Marketing dataset		
Encoding	Before encoding	After encoding
One-hot	17 features	53 features
Deep-learned embedding	17 features	46 features
Adult Income dataset		
Encoding	Before encoding	After encoding
One-hot	15 features	106 features
Deep-learned embedding	15 features	105 features
In-vehicle Coupon dataset		
Encoding	Before encoding	After encoding
One-hot	26 features	100 features
Deep-learned embedding	26 features	80 features

The number of the generated features depends on the number of categorical variable in the original dataset. Before encoding, our original datasets had 17 columns, 15 columns, and 26 columns for the Bank marketing dataset, Adult income dataset, and In-vehicle coupon dataset respectively. Therefore, we have only applied the deep-learned embedding technique on the categorical data and after encoding, in view of the the Bank marketing dataset, the one-hot encoding generated 46 features and our technique generated 39 feature. Here, note that after encoding we have to concatenate all the data that means we have to put together the categorical data encoded and the numerical data. Accordingly, we had 46 (39 + 7) features for deep-learned embedding and 53 (46 + 7) for one-hot encoding as shown in Table 7. Once again our deep-learned embedding is better than one-hot encoding because it generates fewer features than one-hot encoding. Additionally, when using the adult income dataset, our technique generates 105 features and one-hot encoding 106 features; moreover, it generates 80 features less than 100 for one-hot encoding when using the In-vehicle coupon dataset. Overall, our deep-learned embedding generated fewer features than one-hot encoding as described in Table 7. The most important point to keep in mind is that the number of features created is determined by the dataset we are working with.

There are two ways to find how pandas dataframe use the memory, one way is to find the memory usage of each column in bytes in a dataframe and the other way is to find the total memory usage of a dataframe. Therefore, as mentioned earlier, we have to convert the sparse matrix into a data frame in order to find the memory usage. As shown in Fig. 11, our technique uses less memory than one-hot encoding; we can confirm that the memory usage by the dataframe depends on the number of features generated.

When we use deep-learned embedding, first it takes up less memory as shown in Table 8. In view of the Bank

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3104357, IEEE Access

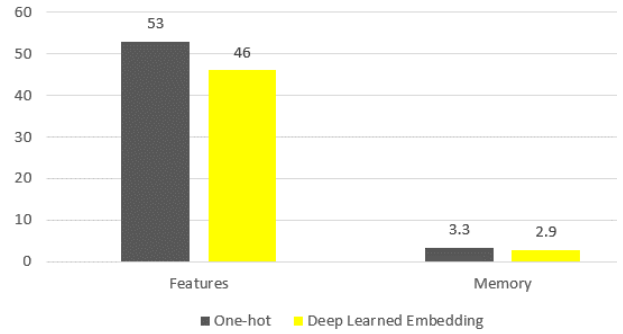

FIGURE 11. Comparison of Generated features between One-hot and Deep-learned embedding as well as memory usage.

Marketing dataset, one-hot encoding uses 3.3 MB while deep-learned embedding uses 2.9 MB. In this paper We also compare these memory usages. With respect to one-hot, the absolute difference is 0.4 MB. That means, if deep-learned embedding needs 2.9 MB to run then one-hot encoding will need 2.9 MB and more to run (2.9 + 0.4 = 3.3), and we can say that one-hot need 13% of memory of deep-learned embedding (0.4 MB equal 13% of 2.9 MB).

TABLE 8. Memory usage

Bank Marketing dataset		
Encoding	Before encoding	After encoding
One-hot	7.2 MB	3.3 MB
Deep-learned embedding	7.2 MB	2.9 MB
Adult Income dataset		
Encoding	Before encoding	After encoding
One-hot	3.91 MB	27.61 MB
Deep-learned embedding	3.91 MB	21.88 MB
In-vehicle Coupon dataset		
Encoding	Before encoding	After encoding
One-hot	2.05 MB	10.174 MB
Deep-learned embedding	2.05 MB	6.494 MB

Furthermore, when using the adult income dataset and encode the categorical variable using our technique, the data frame uses 21.88 MB; however, when we encode with one-hot encoding, it uses 27.61 MB of memory. In addition, using our technique, the memory usage is 6.494 MB less than 10.174 MB for one-hot encoding when taking into account the In-vehicle coupon dataset. Importantly, our deep-learned embedding technique used less memory than one-hot encoding.

V. LIMITATIONS OF THE STUDY

Our study has some limitations within which our findings need to be interpreted carefully. Some limitations of the study should be mentioned. First, as in most empirical studies, the research presented here was limited by the datasets used. Because datasets are composed of categorical values and numerical values. Some categorical values can be a single word (e.g. high, low) or open compound word separated by a space (e.g. high school, graduate school) or hyphenated compound word (e.g. one-half, seventy-two). Second, we focused on binary classification problems in our research. Third, our study did not examine the impact of embedding a compound word. Last but not least, the results of the study may not be completely generalizable because the data was restricted to the purely categorical dataset.

VI. CONCLUSION

Many machine learning algorithms can support categorical values without further manipulation but there are many more algorithms that do not. Therefore, the analyst is faced with the challenge of figuring out how to turn these text attributes into numerical values for further processing. There are many ways we can encode these categorical variables as numbers and use them in an algorithm. This paper demonstrated and compared the classification accuracy and other metrics of machine learning models applied to categorical data encoded using different encoding techniques. The study aims to find a new way to encode categorical variables that can be nominal or ordinal data based on the word embedding technique. It is possible to convert a categorical variable into numeric form even if there is a semantic context or not between categorical data. The goal of this study was to find out the new encoding technique for a categorical variable on a categorical dataset such as the Bank Marketing dataset, Adult Income dataset, and In-vehicle coupon recommendation dataset. Overall, the proposed deep-learned embedding techniques outperform the best results compared to one-hot encoding techniques, and it can use less memory and generate fewer features than one-hot encoding. According to prediction results, our deep-learned embedding technique outperform the best accuracy of 73% and precision of 0.83 compare to one-hot encoding technique, in consideration of the Bank Marketing dataset. This is not an exhaustive study, the deep-learned embedding technique can be considered more preferable for prediction tasks involving purely categorical data.

REFERENCES

[1] Dealing with categorical features in machine learning. accessed 2019. [online]. Available at https://www.kdnuggets.com/2019/07/categorical-features-machinelearning.html.
[2] Tomas Mikolov, G.s Corrado, Kai Chen, and Jeffrey Dean, "Efficient estimation of word representations in vector space," pp 1–12, 01 2013.
[3] Alakh Sethi. One-hot encoding vs. label encoding using scikit-learn. accessed 2019. [online]. Available at https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/.
[4] Sarinnapakorn K. Kuruppu-Appuhamilage I. Chen S.C. Chang L. Goldring Shyu, M.L. and T., "Handling nominal features in anomaly intrusion detection problems,0-8," 15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications.
[5] Kilian Weinberger, Anirban Dasgupta, Josh Attenberg, John Langford, and Alex Smola, "Feature hashing for large scale multitask learning," 2010.
[6] François Chollet,"Deep Learning with Python," Manning Publications, 1st edition, 2017.
[7] Market Research Guy, Types of data and measurement scales: Nominal, ordinal, interval and ratio. accessed 2020. [online]. Available at https://www.mymarketresearchmethods.com/ types-of-data-nominal-ordinal-interval-ratio.
[8] Baijayanta Roy, All about categorical variable encoding. accessed 2020. [online]. Available at https://towardsdatascience.com/ all-about-categorical-variable-encoding-305f3361fd02.

[9] What is the difference between ordinal, interval and ratio variables. accessed 2019. [online]. Available at https://www.graphpad.com/support/faq/what-is-the-difference-between-ordinal-interval-and-ratio-variables-why-should-i-care/

[10] Baijayanta Roy. Type of categorical encoding. accessed 2020. [online]. Available at https://www.analyticsvidhya.com/blog/2020/08/ types-of-categoricaldata-encoding/.

[11] Daniele Micci-Barreca, "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems," SIGKDD Explorations, 3:27–32, 07 2001.

[12] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria, "Recent trends in deep learning based natural language processing," IEEE Computational Intelligence Magazine, 13(3):55–75, 2018.

[13] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa, "Natural language processing (almost) from scratch," CoRR, abs/1103.0398, 2011.

[14] Xuezhe Ma and Eduard H. Hovy, "End-to-end sequence labeling via bidirectional lstm-cnns-crf," CoRR, abs/1603.01354, 2016.

[15] M. Schuster and K.K. Paliwal, "Bidirectional recurrent neural networks," IEEE Transactions on Signal Processing, 45(11):2673–2681, 1997.

[16] Abdulrahman Almuhareb, Waleed Alsanie, and Abdulmohsen AlThubaity, "Arabic word segmentation with long short-term memory neural networks and word embedding," IEEE Access, 7:12879–12887, 2019.

[17] Yan Shao, "Cross-lingual word segmentation and morpheme segmentation as sequence labelling," CoRR, abs/1709.03756, 2017.

[18] Jie Yang, Yue Zhang, and Fei Dong, "Neural word segmentation with rich pretraining," CoRR, abs/1704.08960, 2017.

[19] Cohen P. West-S.G. Aiken L.S. Cohen, J., "Applied multiple regression/correlation analysis for the behavioral sciences," (3rd ed.). routledge. 2002.

[20] Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou, "Word embeddings quantify 100 years of gender and ethnic stereotypes," Proceedings of the National Academy of Sciences, 115, 11 2017.

[21] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin, "A neural probabilistic language model," J. Mach. Learn. Res., 3:1137–1155, March 2003.

[22] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov, "Enriching word vectors with subword information," CoRR, abs/1607.04606, 2016.

[23] Qian Liu, Heyan Huang, Yang Gao, Xiaochi Wei, Yuxin Tian, and Luyang Liu, "Task-oriented word embedding for text classification," In Proceedings of the 27th International Conference on Computational Linguistics, pages 2023–2032, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[24] Jeffrey Pennington, Richard Socher, and Christopher D Manning, "Glove: Global vectors for word representation," In EMNLP, volume 14, pages 1532–1543, 2014.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean, "Distributed representations of words and phrases and their compositionality," In Neural and Information Processing System (NIPS), 2013.

[26] Kedar Potdar, Taher S. Pardawala, and Chinmay D. Pai, "A comparative study of categorical variable encoding techniques for neural network classifiers," International Journal of Computer Applications, 175(4):7–9, Oct 2017.

[27] Wlodzislaw Duch, Karol Grudzinski, and G. Stawski, "Symbolic features in neural networks," In In Proceedings of the 5th Conference on Neural Networks and Their Applications, pages 180–185, 2000.

[28] Krzysztof Grabczewski and Norbert Jankowski, "Transformations of symbolic data for continuous data oriented models," In ARTIFICIAL NEURAL NETWORKS AND NEURAL INFORMATION PROCESSING – ICANN/ICONIP 2003, pages 359–366. Springer, 2003.

[29] Mei-Ling Shyu, Kanoksri Sarinnapakorn, Indika Kuruppu-Appuhamilage, Shu Ching Chen, LiWu Chang, and Thomas Goldring, "Handling nominal features in anomaly intrusion detection problems," In J. Han and H. Kawano, editors, Proceedings of the IEEE International Workshop on Research Issues in Data Engineering, pages 55–62, October 2005. Conference date: 03-04-2005 Through 04-04-2005.

[30] G. Zhong, Lina Wang, and Junyu Dong, "An overview on data representation learning: From traditional feature learning to recent deep learning," ArXiv, abs/1611.08331, 2016.

[31] Brett Lantz, "Machine Learning with R," Packt Publishing, 2013.

[32] Alexander von Eye and Clifford C. Clogg, "Categorical Variables in Developmental Research: Methods of Analysis," Academic Press; 1st edition (February 2, 1996), 1996.

[33] Khoshgoftaar T.M. Hancock, J.T., "Survey on categorical data for neural networks," J Big Data 7, 28 (2020).

[34] Cheng Guo and Felix Berkhahn, "Entity embeddings of categorical variables," CoRR, abs/1604.06737, 2016.

[35] Yixuan Ma and Zhenji Zhang, "Travel mode choice prediction using deep neural networks with entity embeddings," IEEE Access, 8:64959–64970, 2020.

[36] Baohua Sun, Lin Yang, Wenhan Zhang, Michael Lin, Patrick Dong, Charles Young, and Jason Dong, "Supertml: Two-dimensional word embedding and transfer learning using imagenet pretrained CNN models for the classifications on tabular data," CoRR, abs/1903.06246, 2019.

[37] Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou, "Word embeddings quantify 100 years of gender and ethnic stereotypes," CoRR, abs/1711.08412, 2017.

[38] Baohua Sun, Lin Yang, Michael Lin, Charles Young, Patrick Dong, Wenhan Zhang, and Jason Dong, "Supercaptioning: Image captioning using two-dimensional word embedding," CoRR, abs/1905.10515, 2019.

[39] Yoan Russac, Olivier Caelen, and Liyun He-Guelton, "Embeddings of categorical variables for sequential data in fraud context," In International Conference on Advanced Machine Learning Technologies and Applications, pages 542–552. Springer, 2018.

[40] Patricio Cerda and Gaël Varoquaux, "Encoding high-cardinality string categorical variables," CoRR, abs/1907.01860, 2019.

[41] Yushi Yao and Zheng Huang, "Bi-directional LSTM recurrent neural network for chinese word segmentation," CoRR, abs/1602.04874, 2016.

MWAMBA KASONGO DAHOUDA received the BS in information system engineering from University Protestant of Lubumbashi, Lubumbashi, DR Congo, and the MS in software engineering from Hanyang University, Seoul, Korea in 2020. He is currently pursuing the Ph.D. in software engineering at Hanyang University, Seoul, Korea. His research interests include artificial intelligence, deep learning, wireless powered communication networks, and non-terrestrial network.

INWHEE JOE received his BS and MS in electronics engineering from Hanyang University, Seoul, Korea, and his Ph.D. in electrical and computer engineering from Georgia Institute of Technology, Atlanta, GA in 1998. Since 2002, he has been a faculty member in the Department of Computer Science at Hanyang University, Seoul, Korea. His current research interests include Internet of Things, cellular systems, wireless powered communication networks, embedded systems, network security, machine learning, performance evaluation.