Tabular Data Science Final Project - A Numerical in a Haystack: Detecting False Numerical Columns

Gil Or, Dani Bekman

February 2023

1 Abstract

This project aims to enhance the feature engineering phase of the supervised data science prediction pipeline. Our goal is to develop a classifier that can accurately identify whether a numeric column in a given dataset is categorical, and if so, whether it is nominal or ordinal. This information is crucial for determining the best approach for handling the column. The user provides a dataset and target column for prediction. We then train a neural network to extract vector embeddings for potentially categorical numeric columns. By comparing the similarity of the original values to the distances between values in the embedding space, we can identify the type of categorical data represented by the column. Our tool helps data scientists make better decisions about how to prepare their data for model training. We tested our model on four datasets of binary classification and regression tasks and achieved an accuracy score of 94% in predicting whether a categorical variable with a numeric format is ordinal or nominal.

2 Problem description

In some datasets, numerical columns may have a small number of unique integer values and therefore they may be considered as categorical variables. It is essential to properly handle those categorical data in the supervised data science prediction pipeline. One crucial aspect of the supervised data science prediction pipeline is the appropriate handling of categorical columns. It is essential to differentiate between nominal and ordinal data during model training, as this information can impact the model's generalization performance. Typically, nominal variables are converted into one-hot encoding, while ordinal variables are mapped to integer values. The mapped representation establishes an order between the variable values. Therefore, applying mapping to nominal variables may introduce an erroneous and misleading order which might impair the model's results. However, determining whether a given numeric column is categorical and, if so, whether it is nominal or ordinal, can be challenging and may depend on the scientist's domain knowledge.

To address this issue, our tool first detects the numeric columns that are in fact categorical and then it provides a numeric analysis of theses columns. This analysis helps data scientists determine the type of categorical data represented in these columns. By providing more objective and data-driven insights into this question, our tool aims to improve the quality and reproducibility of the DS pipeline, enabling more accurate and reliable model training.

3 Solution overview

To address the challenge of identifying whether a given numeric column is categorical, ordinal or nominal, we developed a solution that combines classification and neural network techniques. Our goal was to create an efficient and robust method for identifying categorical data in numeric columns, which is crucial for successful feature engineering in the data science pipeline.

Our solution comprises three main steps: first, acquiring candidate columns for classification; second, creating embeddings for the values in these columns; and third, comparing the order of the original values to the order in the created embedding space. By using this approach, our tool can accurately identify categorical data in numeric columns and help data scientists perform feature engineering more effectively.

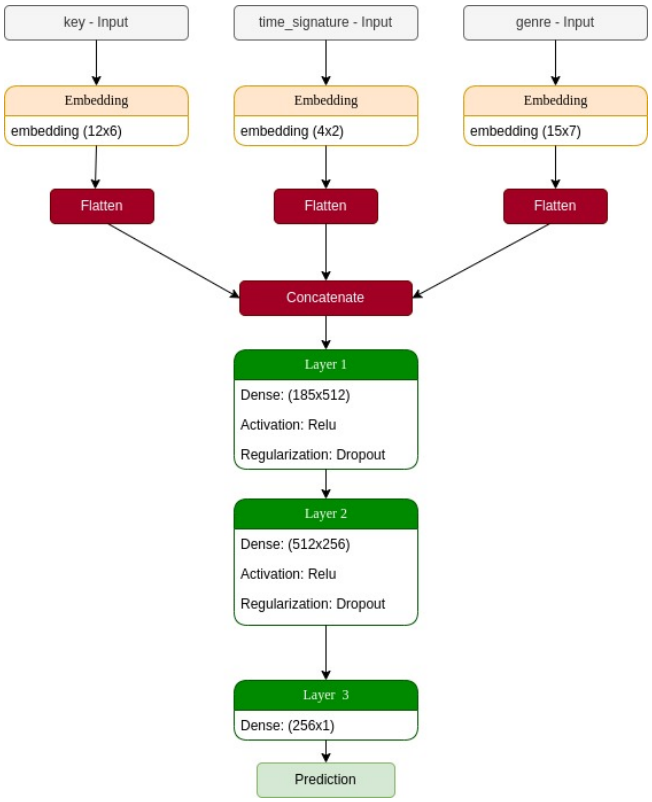
Acquiring Candidate Columns for Classification: The first step of our solution involves examining all columns in the given dataset and identifying those that are originally classified as numeric but should actually be considered categorical. To determine which numeric columns are categorical, we established two classification criteria: the specific numeric type of the column (such as float or integer) and the number of unique values. We found that if a column contains integer values and has fewer than 50 unique values (which can be modified by the user), it should be treated as categorical and passed on to the next phase to determine whether it is ordinal or nominal.

Creating Embeddings for the Values of the Candidate Columns: In the second phase of our solution, we create an embedding representation for each candidate column, allowing us to compare the order of the original values with the order of the vectors in the embedding space. To create these embeddings, we trained a neural network on the original prediction task, using only the columns that required classification. We then extract a dedicated set of network parameters to serve as the column's value embedding (see Figure 1). It is worth noting that if we encounter a regression problem, our architecture converts it to a binary classification task by comparing each value to the mean. This approach allows us to continue using our solution to identify categorical data in numeric columns, even in regression problems.

comparing the order of the original values to the order in the created embedding space: The original idea is to compare the order of the original values to the order of the embedding space in each candidate column. If the order in the embedding space is correlated with the order of the original values, we can conclude that there is a semantic meaning to the values, and the column is ordinal; otherwise, it is nominal. It is not clear how to define a measure of order in a vector space, to avoid this issue, we defined a unique measure to compare the orders. We created distance matrices for both the original space and the embedding space, with each cell representing the distance between values i and j in the column. These matrices were N x N, where N is the number of unique values in the candidate column. To measure the degree of similarity between the distance matrices, we calculated the mean Spearman correlation between the rows. We then compared the resulting mean correlation value to a threshold of 0.4 to determine whether the column should be classified as ordinal or nominal. If the mean correlation score is 0.4 or higher, the column is defined as ordinal; otherwise, it is nominal. This threshold was set by a general understanding of the strength of a correlation between two arrays. generally, two arrays with correlation coefficient of 0.4 or higher is considered to indicate moderate to high similarity between the two arrays[1]. It is worth noting that the user can modify this decision threshold and the user also receives the correlation score that led to a specific decision, which can provide the level of certainty of the model.

Our solution allows us to get insight about a categorical column classification in interactive time and provides an efficient and generalizable method for identifying categorical data in numeric columns, and can be applied to a wide range of datasets and modeling tasks. By using our approach, data scientists can make more informed decisions about feature engineering and model training, leading to more accurate and reliable results.

Figure 1: The network architecture used to get vector embeddings for dummy variables



The size of the embedding matrices is derived by: #rows = #unique values in the column. #column=$\frac{\#unique}{2}$. Each row in the matrix represents a vector of a value in the original input. The inputs to the network are only the columns that we need to check.

4 Experimental evaluation

We tested our model on four datasets: the Adult dataset, the Spotify dataset, the Titanic dataset, and the Video Games dataset. The latter involves a regression prediction task, while the others are binary classification prediction tasks. To aid our classification efforts, we generated additional data by converting string values in categorical columns to random or ordered numeric values. We converted categorical columns with known classifications to integers, using a pre-defined order for ordinal columns, and random mappings for nominal columns. For example, the Education column in the Adult dataset contains values such as "1st-4th", "5th-6th", "7th-8th", "9th", "10th", "11th", "12th", "Assoc-acdm", "Assoc-voc", "Bachelors", "Doctorate", "HS-grad", "Masters", "Preschool", "Prof-school", and "Some-college". We mapped these values to a numeric scale using a pre-defined order based on our world knowledge. For instance, "1st-4th" was mapped to 0, "5th-6th" to 1, and "Doctorate" to 15 (which is the total number of unique values in the column). The Workclass column in the Adult dataset contains values such as "Private", "Self-emp-not-inc", "Self-emp-inc", "Federal-gov", "Local-gov", "State-gov", "Without-pay", "Never-worked". As to our world knowledge, this values do not hold order, therfore we mapped it randomly. The exhaustive details of the data manipulations we performed are available in the dataset folder of our repository. After applying the data manipulations, we had 17 columns to test, including 7 ordinal and 10 nominal columns. The original and converted datasets are available in the dataset folder of our repository.

We defined a baseline approach that involved converting the columns to check into dummy variables, and defining them as X for a linear regression model. We defined Y as the user-provided target variable, and ran a simple linear regression between X and Y. The output of this analysis was a list of Beta Coefficients for each dummy variable. We calculated the Spearman correlation between the original order of the values and their corresponding Beta Coefficients. This method achieved an accuracy score of 0.75, as shown in Table 1.

As shown in table 1, our solution achieved an accuracy score of 94%, successfully predicting 16 of the 17 tested columns. Our model miss-classified the Embarked column of the Titanic dataset. This column represents the port that a passenger embarked at, we assumed it should be nominal but the model classified it as ordinal. further review of this column statistics showed that more than 70% of the data originated in a single value. This may explain why our model struggled with predicting the column type. Future work can be done in order to handle this type of skewed columns.

Table 1: Comparison table between baseline regression and nueral network

Dataset	Task type	Column name	Gold	Linear regression result	Neural Net result
Adult	binary classification	workclass	Nominal	Nominal	Nominal
Adult	binary classification	marital status	Nominal	Nominal	Nominal
Adult	binary classification	occupation	Nominal	Nominal	Nominal
Adult	binary classification	native-country	Nominal	Nominal	Nominal
Adult	binary classification	relationship	Nominal	Nominal	Nominal
Adult	regression	education	Ordinal	Ordinal	Ordinal
Adult	regression	educational num	Ordinal	Ordinal	Ordinal
Spotify	regression	key	Nominal	Ordinal	Nominal
Spotify	regression	genre	Nominal	Nominal	Nominal
Spotify	regression	time signature	Ordinal	Ordinal	Ordinal
Titanic	binary classification	SibSB	Ordinal	Nominal	Ordinal
Titanic	binary classification	Pclass	Ordinal	Nominal	Ordinal
Titanic	binary classification	Embarked	Nominal	Nominal	Ordinal
Titanic	binary classification	Parch	Ordinal	Ordinal	Ordinal
Video Games Sales	regression	Year of Release	Ordinal	Ordinal	Ordinal
Video Games Sales	regression	Platform	Nominal	Nominal	Nominal
Video Games Sales	regression	Genre	Nominal	Nominal	Nominal

The accuracy for the linear regression baseline is 0.7647. The accuracy for the neural network is 0.94116.

5 Related work

As far as we know there are no prominent studies that address this issue. There are several papers about data type detection [3]. Our solution is different from this for two reasons. First, in data type detection, the set of types is limited to the types of a given knowledge base. For example, the Titanic dataset we used for evaluating our model includes the Parch column, which means number of parents and off-springs on the Titanic board. We assume that there will not be a matching entity to the Parch variable in any knowledge base. Our solution does not require a knowledge base for its prediction. Second, in data type detection, after detecting the predicted type, it is still up to the user to decide whether the type is ordinal or nominal. Our solution automatically decides on the type. The idea of using the embeddings as a by-product of the standard supervised neural network training process came from [2].

6 Conclusion

The goal of this project was to improve the feature engineering phase of the supervised data science prediction pipeline by developing a tool to accurately identify whether a numeric column in a given dataset is categorical and, if so, whether it is nominal or ordinal. The proposed solution combines classification and neural network techniques and comprises three main steps: candidate column acquisition, embedding creation, and comparison of the order of original values with the order of vectors in the embedding space. The results showed that the tool achieved an accuracy score of 94% in predicting whether a categorical variable with a numeric format is ordinal or nominal, based on testing on four datasets of binary classification

and regression tasks. By providing more objective and data-driven insights into the identification of categorical data in numeric columns, this tool can help data scientists make more informed decisions about feature engineering and model training, leading to more accurate and reliable results.

References

[1] Spearman's correlation. Available online: http://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf. (accessed on 28 February 2023).

[2] Cheng Guo and Felix Berkhahn. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*, 2016.

[3] Michiel Hulsebos, Ke Hu, Marten Bakker, Emanuel Zgraggen, Arun Satyanarayan, Tim Kraska, et al. Sherlock: A deep learning approach to semantic data type detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1500–1508. ACM, 2019.