Evaluating Dense-Binary Encoding for Regression

Jillian Anderson
Department of Computer Science
Simon Fraser University
jilliana@sfu.ca

Shawn Anderson
Department of Computer Science
Simon Fraser University
shawn_anderson@sfu.ca

Brie Hoffman
Department of Computer Science
Simon Fraser University
brieh@sfu.ca

Ka Hang Jacky Lok
Department of Computer Science
Simon Fraser University
khlok@sfu.ca

Abstract

Encoding categorical features as numeric values is an important step in regression. One-hot encoding, a commonly used method, adds an extra boolean column for each unique value of categorical variables, significantly increasing the number of feature columns in a dataset. This causes problems for algorithms negatively impacted by increased dimensionality and for large datasets where increases to memory can be prohibitively expensive. We propose dense-binary encoding as a memory efficient alternative for encoding categorical variables. We compare the encoding methods across eight learning algorithms to predict housing prices. Results indicate dense-binary encoding may offer a more memory-efficient encoding method, with comparable accuracy for regression problems.

1 Introduction

As with other machine learning tasks, regression often requires preprocessing of input data. Preprocessing is an important and time-consuming aspect of machine learning which significantly impacts how learning algorithms perform [2, 5]. Many regression models require transforming categorical features to continuous values. This preprocessing task is called encoding. There are three fundamental types of categorical features: boolean, discrete ordered (ordinal), and discrete unordered (discrete). Boolean features represent the presence or absence of an attribute and can be naturally encoded to the domain $\{0, 1\}$. Ordinal features can be naturally encoded to a set of ordered integers. This paper focuses on unordered discrete features, which do not have a natural mapping to a continuous set. Encoding henceforth refers to the encoding of unordered discrete features.

Encoding is typically performed using one-hot encoding (OHE), which uses a sparse binary representation to encode categorical features [9]. Despite its common use, researchers have indicated OHE's propensity for adding features causes issues associated with dimensionality [9]. For example, the addition of features to a model often results in significant decreases to the model's predictive power, a phenomenon called the "Curse of Dimensionality" [10]. However, alternative encoding methods can be used to address the problems associated with OHE.

Dense-binary encoding (DBE), is an alternative encoding method which reduces the number of extra features added to the dataset. In DBE, the categorical variables are first randomly mapped to consecutive integer values. These values are then converted to a binary representation, where each digit becomes its own feature in the dataset. While demonstrations have shown DBE to be a viable solution for classification problems, little research exists on how it holds up during regression [11]. We believe that by reducing the number of features created by encoding, DBE offers a promising alternative to OHE, especially in the context of big data.

2 Approach

For our project we created two datasets by encoding the discrete variables using OHE and DBE. Next, we applied eight popular machine learning models, and evaluated the results. We used root-mean-squared-logarithmic error to evaluate accuracy between encoding methods and models.

2.1 One-Hot Encoding

One-hot encoding (OHE) is performed by creating a new feature of binary values (0, 1) for each possible value from the original categorical column. For each discrete feature C with values $i \in \{x_0, \ldots, x_{n-1}\}$, n new features are created and labeled $C_0, \ldots, C_{n-1}$. For each record where $C = x_i$, features $C_0, \ldots, C_{n-1}$ are set to 0, except for C_i which is set to 1. OHE adds N new features to the dataset, where N is the sum of unique values for all discrete features.

2.2 Dense-Binary Encoding

Dense-binary encoding (DBE) offers an alternative to the OHE method. For each discrete feature C with values $i \in \{x_0, \ldots, x_{n-1}\}$, m new features are created where $m = \lceil \log_2 n \rceil$. The features are labeled $C_1, \ldots, C_m$. For each record where $C = x_i$, x_i is mapped to a base-ten ordinal value such that $x_i \rightarrow i$. Each value i is mapped to the vector of binary values $\vec{b} = [b_0, \ldots, b_{m-1}]$, where each component is a coefficient b_j such that:

$$i = \sum_{j=0}^{m} b_j 2^j$$

A potential problem with Dense-Binary Encoding is that it introduces a relationship between the values of a particular feature. Since the binary code for one value will share more bits with some values than with others, the values unintentionally become "more" or "less" similar. We believe this consequence can be overcome by randomly assigning ordinal values to categorical values. Since the probability of 2 bits matching between any two bit codes is $p = 0.5$, the Hamming distance between any two binary codes will follow a binomial distribution with mean centered at np where n is the length of the bit string [8].

2.3 Root Mean Squared Logarithmic Error

We used the root mean square logarithmic error (RMSLE) metric to evaluate the predictions of each of our models. Where n is the number of observations, p_i is the prediction of i, and a_i is the true value of i, RMSLE is defined as:

$$RMSLE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\ln(p_i + 1) - \ln(a_i + 1))^2}$$

3 Experiments

As shown above, DBE provides superior feature dimensionality and memory efficiency. However, to determine how the two encoding methods compare in accuracy, we conducted three experiments on a single dataset in the context of house price regression. Each experiment uses the same eight regression models and compares performance across encodings.

Our first experiment compared OHE and DBE by applying regression models with default hyper-parameter settings. Experiment 2 compared OHE and DBE by applying regression models with tuned hyperparameter settings. In our final experiment, we combined the tuned regression models to create an ensemble model for each encoding method and compared their resulting errors.

Figure 1 shows the experimental work-flow. We began by performing basic pre-processing on the data. Next, we applied DBE and OHE to create two datasets. Finally, we performed three experiments and evaluated the resulting predictions.

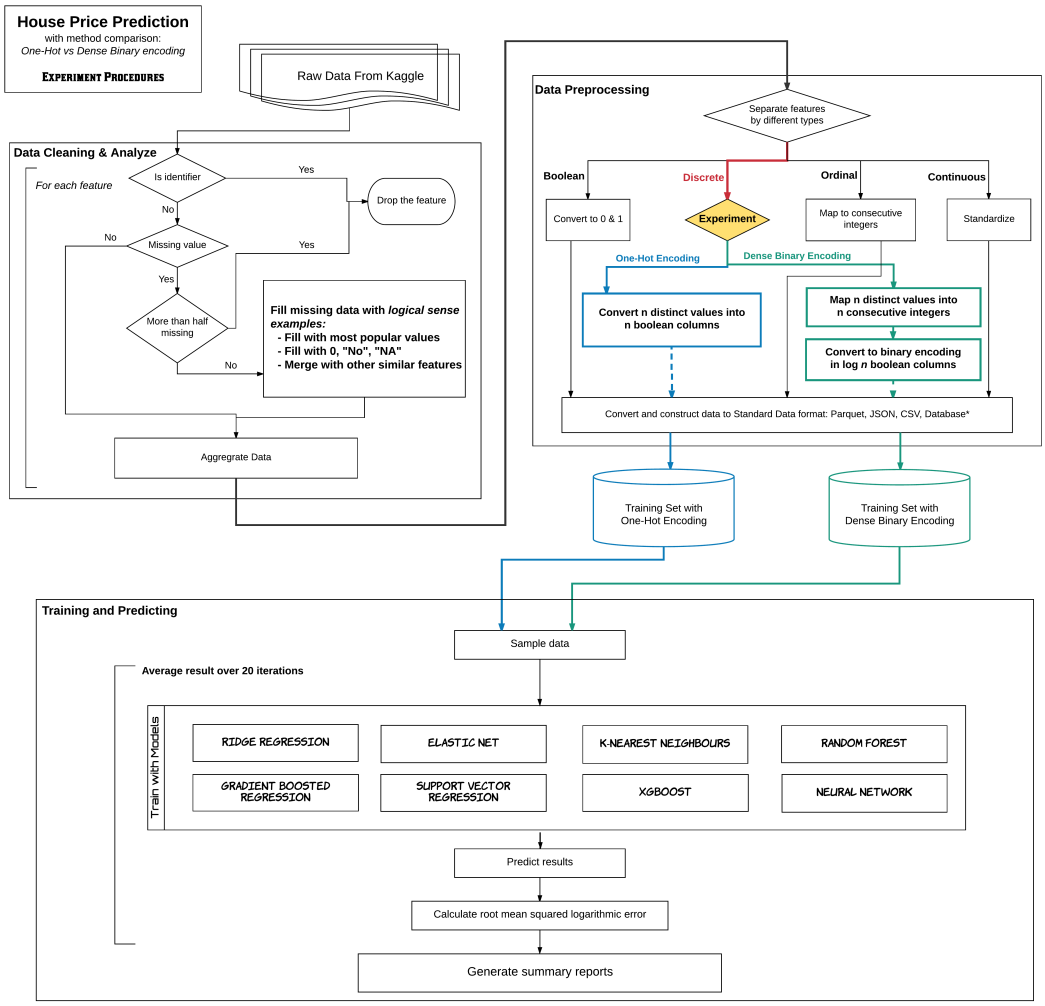

Figure 1: Flowchart describing our approach.

3.1 Data & Pre-Processing

The data used for this project was downloaded from Kaggle's House Prices: Advanced Regression Techniques' competition and was originally compiled by Dean De Cock [3, 4]. The data contained 79 explanatory variables and a single target variable, representing features of 1460 homes sold in Ames, Iowa. This hybrid dataset contained a combination of 35 continuous and 44 categorical features. Of the categorical features, 1 was boolean, 15 were ordinal, and 28 were discrete unordered.

We implemented standard pre-processing as detailed in Sergei Neviadomski's kernel posted on Kaggle [7]. Features which had more than half missing values or did not correlate with sale price were dropped. Missing continuous values were filled in using their respective column's mean. Missing categorical values were filled in using the most common value in their respective column. Continuous columns were standardized to have a mean of zero and a standard deviation of one.

After completing preprocessing, we encoded the remaining categorical features using both OHE and DBE, generating two fully processed datasets. Of the initial 28 discrete unordered columns, 5 were dropped during preprocessing. The remaining 23 discrete unordered columns contained a total of 173 unique values, which for OHE added 173 columns, and for DBE added 68 columns. In the end, the OHE dataset contained 204 columns and the DBE dataset contained 99 columns, a difference of 105 columns, demonstrating the $n \rightarrow logn$ reduction for the 23 discrete unordered features.

3.2 Experiment 1: A comparison of encodings using default parameters

To obtain a baseline comparison between the two encoding methods, we applied eight basic regression models with no tuning, using default parameter settings or guidelines found in documentation. We used two linear models (elastic net, ridge), two non-parametric models (KNN, SVR), one neural network, and three ensemble methods (random forest, GBR, XGBoost). These models were chosen to represent a wide variety of popular models. The following steps were repeated for each model:

1. Create a 75%/25% train/test random split of the OHE train data
2. Train and score the algorithm using baseline parameters
3. Repeat steps 1 & 2 twenty times, to compute an average performance score
4. Repeat steps 1, 2, & 3 with DBE train data, using the **same model parameters**.

3.3 Experiement 2: A comparison of encodings, tuning for best results

Our second experiment considers that a model may have differing optimal hyper-parameter settings depending on how the data is encoded. We reimplemented the process of Experiment 1, but did not fix parameters across encodings. Instead, the parameters were tuned separately for each encoding method to find an approximation of the lower bound error for both OHE and DBE.

3.4 Experiment 3: A comparison of encodings between ensembled algorithms

Finally, we created two ensemble models to compare how OHE and DBE performed overall on the test set. The first ensemble model used the mean of the test-set predictions obtained by each model as its ensembled prediction. The second ensemble model used a weighted average of the test-set predictions obtained by each model as its ensembled prediction. The weights for predictions from each model were computed using:

$$w_m = \frac{1 - min(\epsilon_m, 1)}{\sum_{m \in M} 1 - min(\epsilon_m, 1)}$$

where w_m is the weight assigned to predictions made by model m, ϵ_m is model m's RMSLE on the validation set, and M is the set of all eight models. The inclusion of the min function ensured models with RMSLEs greater than 1 were excluded from the ensemble model. This results in higher weights for predictions produced by models which obtained low errors on the validation set. We then found the RMSLE of the ensemble methods by submitting the resulting test-set predictions to the Kaggle competition.

4 Experimental Results

As shown in Figure 2 and Table 1, we found models using default parameters tended to perform better when used with OHE rather than DBE. The only model which performed better with DBE was the neural network model, where DBE achieved an RMSLE of 0.333 compared to the OHE error of 0.678. Due to the large reductions in error when using DBE with the neural network model, the overall average error was reduced by using DBE. Additionally, in cases where models performed better with OHE than DBE, the differences in errors was relatively small and often only present when considering more than two significant digits.

When using tuned models, we found that OHE consistently outperformed DBE by a small margin, as shown in Figure 3 and Table 1. Models which used DBE had an average RMSLE of 0.165, which is 7.14% greater than the error obtained when using OHE.

Finally, as shown in Table 2, ensemble models which use DBE outperformed those which used OHE. In fact, the DBE unweighted ensemble model outperformed the OHE weighted ensemble model. It should be noted that the figures presented in Table 2 show the RMSLE values obtained by submitting the test-set predictions to the Kaggle Competition. Thus, they should not be directly compared to the values in Table 1, which were obtained using a hold-out validation set.

Figure 2: RMSLE for eight regression models using default parameters

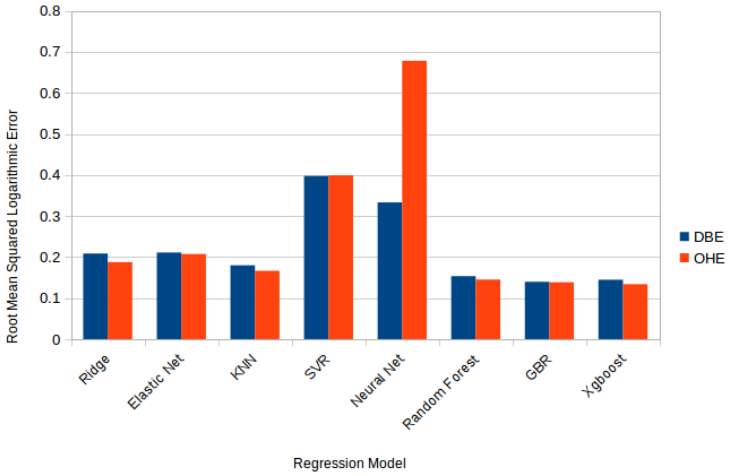

Table 1: Average RMSLE over 20 iterations. The best results for each model type are bolded.

Model	**Default Average Error** DBE	OHE	**Tuned Average Error** DBE	OHE
Ridge Regression	0.209	0.188	0.209	0.165
Elastic Net	0.211	0.207	0.190	0.185
K-Nearest Neighbour	0.180	0.167	0.174	0.168
Support Vector Regression	0.398	0.399	0.177	0.165
Neural Network	0.333	0.678	0.142	0.134
Random Forest	0.153	0.145	0.147	0.144
Gradient Boosted Regression	**0.140**	0.138	**0.138**	0.138
XGBoost	0.145	**0.134**	0.144	**0.132**
Total Error	1.768	2.450	1.320	1.232
Average Error	0.221	0.306	0.165	0.154

Figure 3: RMSLE for eight regression models, tuned for best results

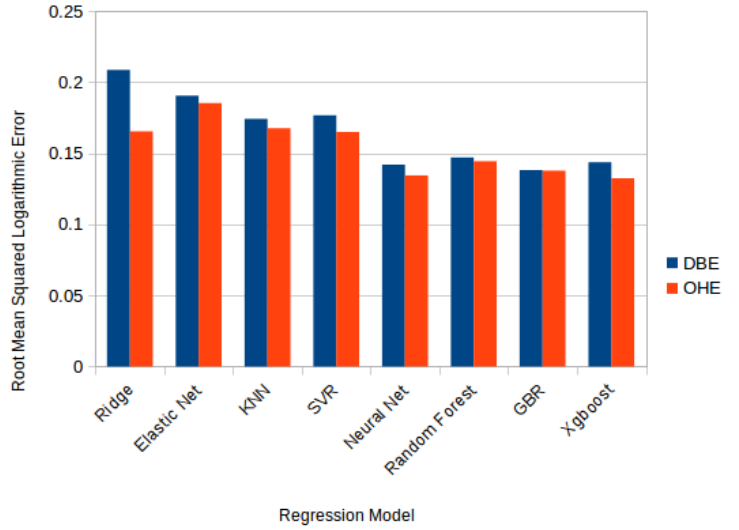

Table 2: Results comparing the test score performance across encodings, given an ensemble of all eight algorithms.

Model	DBE Test Set Error	OHE Test Set Error
Eight Model Mean Prediction	0.13921	0.16609
Weighted Eight Model Mean Prediction	0.13890	0.14949

5 Conclusions

Our results indicate that OHE tends to provide slightly more accurate predictions than DBE when used with individual regression models. However, DBE did outperform OHE when used with an ensemble model. This discrepancy could be because errors for individual models were calculated using the validation set, while errors for the ensemble models were calculated by Kaggle, using the test set. This serves to highlight the high similarity between the accuracies obtained by OHE and DBE. Combining this finding with DBE's ability to reduce feature dimensionality and improve memory efficiency, this makes DBE a viable alternative to OHE.

The possible trade-off of accuracy in favour of dimensionality and memory reduction is well justified when dealing with datasets containing large numbers of high-cardinality categorical features, where DBE would offer significant dimensionality reduction over OHE. In these cases, the ability to reduce dimensions and thus simplify the models may outweigh the small decrease in accuracy associated with using DBE.

Additionally, DBE's reduced accuracy may be justified in the context of big data. It is possible that a dataset may contain a large enough number of rows that the addition of a large number of columns may be prohibitively expensive. In these cases, DBE may offer a reasonably accurate and more viable solution for categorical encoding.

We believe further research should be conducted to determine the effectiveness of DBE in comparison to other preprocessing schemes for categorical attributes such as clustering [1], and categorical encoding using target statistics [6].

Contributions

The four authors contributed equally to this project, but focused on different tasks. The tasks which each author made significant contributions to are shown below:

1. **Jillian Anderson**: Research, data cleaning and encoding, neural net modeling, results analysis, report writing.
2. **Shawn Anderson**: Research, data cleaning and encoding, K-Nearest Neighbours, Xgboost and Random Forest modeling and model tuning, ensemble averaging, report writing.
3. **Brie Hoffman**: Research, Support Vector Regression and Gradient-boosted Trees modeling, Gradient Boosted Tree model tuning, results analysis, report writing.
4. **Ka Hang Jacky Lok**: Research, work-flow diagram, data preprocessing, Elastic Net and Ridge Regression modeling, model tuning for Elastic Net, Ridge Regression, Neural Net and Support Vector Regression, ensemble averaging.

References

[1] Jonathan D Becher, Pavel Berkhin, and Edmund Freeman. Automating exploratory data analysis for efficient data mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–429. ACM, 2000.

[2] CrowdFlower. Data Scientist Report. Technical report, CrowdFlower, San Francisco, 2017.

[3] Dean De Cock. Ames, iowa: Alternative to the boston housing data as an end of semester regression project. *Journal of Statistics Education*, 19(3), 2011.

[4] Kaggle. House prices: Advanced regression techniques. https://www.kaggle.com/c/house-prices-advanced-regression-techniques.

[5] SB Kotsiantis, D Kanellopoulos, and PE Pintelas. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, 2006.

[6] Daniele Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *SIGKDD Explor. Newsl.*, 3(1):27–32, July 2001.

[7] Sergei Neviadomski. How to get to top 25/ https://www.kaggle.com/neviadomski/how-to-get-to-top-25-with-simple-model-sklearn.

[8] Rasmus Berg Palm. Dense codes. http://rasmusbergpalm.github.io/2016/02/12/dense-codes.html.

[9] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 1149–1154. IEEE, 2016.

[10] S Russell and P Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, Upper Saddle River, 3 edition edition, December 2009.

[11] McGinnis W. Beyond One-Hot: incremental improvements in categorical encoding. http://www.willmcginnis.com/2016/06/17/beyond-one-hot-incremental-improvements-categorical-encoding/, June 2016.