**Using Categorical Embedding to Improve Restaurant Review Classifier Predictions  
  
James Otwell**

**May 2024**

**Abstract**

Many Machine Learning classifiers can struggle with prediction using a mixture of categorical and continuous features. One more recently developed approach to addressing this challenge is categorical embedding, a technique that uses Artificial Neural Networks to map categorical features to the Euclidean space. While this approach has been shown to improve Neural Network classification accuracy, there has not been much research into its impact on other Machine Learning classifier algorithms. This paper studied whether categorical embedding could improve the performance of K-nearest neighbors, Random Forest, and Artificial Neural Network at predicting restaurant reviews using a mixture of categorical and continuous features. The experiment found that Random Forest, as expected, performed better as a classifier with traditional one-hot encoding of the features while categorical embedding significantly improved the performance of K-nearest neighbors and Artificial Neural Network. For one of the prediction tasks, Artificial Neural Network even outperformed Random Forest after categorical embedding. The results suggest more research should be conducted into how categorical embedding can improve non-Neural Network Machine Learning classifiers.

**Introduction**

Many datasets have a mixture of categorical and continuous features. While this mixture can be useful for an analyst finding patterns in the data, it can challenge many Machine Learning classifier algorithms. Many classifiers such as K-nearest-neighbors (KNN) and Artificial Neural Networks (ANN) are sensitive to feature scaling, the appropriate method often being difficult to determine when considering continuous and categorical features together [1][2][3]. Additionally, cardinal categorical features require one-hot/ dummy encoding, which, if the features have many categories, can greatly increase the dimensionality of the data space, and create both sparsity and runtime issues[3]. As a result, tree-based methods such as Decision Tree and Random Forest are often favored for handling these datasets (but are not without their own limitations)[3][4][5]. One such technique that is used to address both challenges is Categorical Embedding, which utilizes ANN architecture to map each categorical value to the continuous space[1]. The benefits of these embeddings in comparison to one hot encoding is that they are lower-dimension, theoretically more meaningful, and able to be easily considered with continuous features[1]. Thus, if the embedding is effective, it would theoretically allow more Machine Learning model types to be easily applied with mixed-attribute data. This paper will further research the utility of categorical embedding when preprocessing for ML models.

Research into categorical embedding is more recent and mainly focuses on how it enhances ANN and unsupervised learning performance with mixed data. The technique is less studied on how it impacts the performance of other classifier algorithms such as KNN and Random Forest. This study used a mixed-feature set of restaurant review rankings from Zomato to compare the performance of ANN, KNN, and Random Forest at predicting the review score classification under both traditional one-hot encoding and categorical embedding. Score classification was conducted both as a binary class problem (above or below average), and a 3-class problem (“bad”, “average,” or “good”). The paper tested 3 hypotheses. The first hypothesis was that Random Forest would significantly outperform the other 2 algorithms in predictive accuracy under traditional one-hot encoding. The second hypothesis was that categorical embedding would significantly improve the performance of ANN and KNN (but not Random Forest). The final hypothesis was that after categorical embedding, ANN, KNN, and Random Forest would either not be significantly different in predictive performance or KNN and ANN would surpass Random Forest. All 3 hypotheses were accepted after conducting trials with 5-fold cross-validation tested at a 95% confidence interval. Random Forest outperformed ANN and KNN with one-hot encoding while categorical embedding significantly improved the accuracy of ANN and KNN to the point that KNN was not significantly worse than Random Forest for both binary and 3-class prediction, and ANN performed significantly better for binary prediction and insignificantly different for 3-class prediction.

**Background**

The challenge of handling mixed attribute data, a mixture of categorical and continuous variables, is well documented in literature on Machine Learning. While I will discuss specifically how these challenges show up for both KNN and ANN in subsequent sections, generally, the challenge presented is the basis for numerical comparison between one-hot encoded categorical data and the continuous numerical features [3]. For a continuous feature such as weight, for example, the difference between 100-200 pounds and 200-300 pounds is the same. Because one-hot encoded categories are discretely encoded as 0 or 1 depending on whether an entry belongs to that category, it does not carry the same numerical relationship as the distance is always either 0 or 1 no matter how many classes there are. This further complicates issues of scaling continuous variables, which is important when continuous variables operate across difference scales and ranges. When compared to a continuous feature under standard scaling, belonging to a class would be seen as being equivalent to being 1 standard deviation away from the mean. Under min/max scaling, it would be equivalent to the largest value of the continuous class (neither of which may be necessarily accurate representations of the categorical feature). Thus, many machine learning methods that do well with either all continuous or all categorical features tend to degrade under mixed feature types [3][4]. This is problematic as many real-world datasets are mixed-feature type.

Due to the performance issues of Machine Learning models such as KNN and ANN on mixed attribute data, traditional regression methods and tree-based methods such as Random Forest are typically favored under these circumstances [1][5]. This is because regression can treat categorical features as intercepts, and tree-based methods only view one feature at a time in each node and do not need to take scales into account when finding the decision boundary for each node. Tree-based methods, in particular, are therefore robust to user preprocessing decisions regarding scaling, which are not intuitive in mixed attribute data[5]. However, these methods are not without their drawbacks. In the case of regression, it assumes a linear relationship between predictors and classes, which might not be true and will limit accuracy[1]. In the case of Random Forest, it can struggle when the data contains features that contain a high number of categories, is unbalanced, is a time series relationship, and is sensitive to the choice of parameters chosen by the user[7]. Therefore, creating an effective way to express mixed attribute data would be helpful for allowing more feasible choices for Machine Learning algorithms to use.

*KNN and Mixed Data*

KNN is challenged by mixed attribute data due to both the increase in dimensionality posed by one-hot encoding, and the issue of finding an appropriate scale and distance metric to judge which points are nearest neighbors[2]. KNN is especially susceptible to the curse of dimensionality due to high dimensional data creating a sparse space that creates few truly similar points. Regarding scale, the challenge is two-fold. First, if the variables are not scaled properly, then nearness will be dominated by the largest-scale features[2][6]. Second, appropriate distance measure becomes cloudy because categorical nearness is likely best measured by cosine similarity, but continuous features are likely best measured by one of the Minkowski metrics[6]. Combining these can be challenging, so alternative approaches have been tried.

While I could not find any literature where categorical embedding using ANN architectures were used with KNN, there is some research into turning categorical features into continuous embeddings for KNN. Luo et al. created a value-object hierarchical distance metric that created numerical embeddings for categorical data to be used with KNN[6]. While the metric they found improved performance of KNN with categorical data compared to other metrics tried, it was not studied with mixed data. One approach that was studied with mixed attribute data was conducted by Tuerhong, Wushouer, and Zhang [2]. They utilized PCAmix to create a PCA loading for both the categorical and continuous features to reduce the dimensionality of the data space as well as embed continuous values to the categorical features before running the KNN algorithm. This approach worked better than KNN with traditional embedding for the datasets they studied. However, it is susceptible to some of the other limitations of PCA in that it assumes maximizing variance in the feature space as the embedding. Categorical embedding, as will be discussed later in this section, maps the variables based on their relationship to the prediction target and could perform better.

*ANN and mixed data*

Much like KNN, ANN classifier performance can also struggle with mixed feature data. In addition to the same challenges of scaling and sparsity that impact KNN, there are additional struggles when using generative models in creating a noise schedule for mixed attribute data, which impacts the quality of output from the generator[7]. They also degrade performance of other neural network tasks such as time-series prediction[8], transfer learning[8], and natural language processing [9].

*Categorical Embedding and Improvements in Machine Learning*

Categorical embedding is a technique that uses ANN architectures to assign continuous scores to different categorical feature labels and map them in the Euclidean space[1][8]. It is based on text-embedding strategies, but applied to categorical data[1]. While there may be some nuances to the different methods (such as selecting the number of neurons, regularization, etc.), the basic structure behind the categorical embedding techniques that I found (and the structure of the one used in this study) is a 2-layer ANN that treats the embeddings as weights of categorical features and trains them based on how they impact the target for prediction [1][8][9].

So far, the applications studied for categorical embedding primarily surround their improvement of ANN performance. Sarthank, Shukla, and Tripanthi increased the predictive performance of a Convolutional Neural Network in predicting 30-day hospital readmission rates by using categorical embedding [9]. Dahouda and Joe also found that using categorical embedding improved the performance of a Long Short-Term Memory network in classifying multiple time series datasets compared to one-hot encoding [8]. Finally, Ahn, Ko, and Kang used regularized categorical embedding to improve bike sharing demand classification[1]. In this last case, the authors found that by using categorical embedding, they were better able to predict bike sharing by bike station as opposed to only being able to predict global patterns with one-hot encoding.

There have been some other studies about other uses for categorical embedding beyond ANN improvement. Li et al. used categorical embedding to increase regression accuracy of various tree-based methods in predicting cancer clinical endpoint[4]. In articles by Shi and Shi and Ahn, Ko, and Kang the authors found that categorical embedding allowed for effective use of Principal Component Analysis, and K-means cluster, respectively, on mixed-attribute data[10][1]. Finally, in comparing ANN performance to other methods, Dahouda and Joe found no improvement in Random Forest classification accuracy with categorical embedding[8].

Besides the previously mentioned article that discusses Random Forest classifier performance, I could not find research on categorical embedding’s impact on other Machine Learning classifiers such as KNN, Support Vector Machines, and Logistic Regression. Beyond ANN prediction, most of the focus I found was on unsupervised learning. This further justifies the significance of evaluating how categorical embedding might improve KNN performance.

**Methodology**

*Data Description*

This experiment used restaurant review data from Zomato. It consisted of 7105 restaurants in and around Bangalore India. The main target for prediction was the average review score on a scale of 1 to 5. The dataset contained the restaurant name, restaurant address, an index column for every restaurant, and 7 total features.

The 2 continuous features of the dataset were the average cost of dining for 2 people and number of reviews. These features were, for the most part, normally distributed, but with a few outliers. Cost ranged from 200 to 6000 in the case of cost, and number of ratings ranged from 6 to 16000. The target class for prediction, average rating out of 5 ranged from 1.8 to 5. These scores appeared normally distributed around the mean value of 3.51 (see Appendix figure 1).

Of the 5 categorical features, there were two binary categorical features: whether the table has online ordering, and whether the diners can book a table ahead of time. About half of the restaurants had online ordering, and about 10% of the restaurants allowed reservations. At a glance, online ordering seemed evenly represented in the positive and negative reviews while an overwhelming majority of the restaurants that took reservations were positively reviewed (see appendix figures 2a and 2b), There were 3 multi-category features: area, cuisine type, and restaurant type. The details of these features will be further examined when discussing the one-hot encoding preprocessing.

*Data Preprocessing*

First, the indexing, restaurant names, and address columns were eliminated. There were also a small number of missing values in the dataset, all in either the rating column or the average cost of 2 diners column. In the case of the latter, the mean cost by restaurant type was imputed. In the case of missing scores, these restaurants were omitted from the dataset since it was the target for prediction.

Continuous features were scaled using standard scaling as they were, for the most part, normally distributed. I chose to keep outlier values of number of reviews and cost as these tended to positively associate with class presence consistent with the rest of the data, and likely would not confound the predictors. Online ordering and table booking were converted from True/False to 1/0. The remaining categorical features will be discussed in the next section.

Finally, before conducting the encoding experiments, I converted the numerical rating score from 0 to 5 to binary and 3-class targets. For binary, the target was coded 0 if the rating was 3.5 or below, and 1 if the data was above 3.5 (the mean). This resulted in an approximately 46/54% positive/negative class balance. For the 3-class problem, I did analyzed the tertile distribution and converted scores to “good” if they were above 3.7, “bad” if they were below 3.3, and “average” if they fell between. This resulted in a 36/32/32% split between bad/average/good.

*One-hot Encoding*

Of the 3 features that required one-hot Encoding, the most straightforward one was “area,” which was a simple string which had 30 categories. Each area had at least 20 restaurants in it with most having at least 100 (see appendix figure 3). The more challenging features that required an intentional decision about one-hot encoding were “restaurant type” and “cuisine type.” What was particularly challenging about these features was that they were expressed as comma delimited strings all in one column, were of inconsistent length, and not consistently alphabetized. For example, one restaurant might have cuisine types “South Indian, North Indian”, another “North Indian, South Indian,” and a third “South Indian, North Indian, Mediterranean.” Under traditional one-hot encoding, all 3 of these restaurants would be categorized being completely different even though all 3 serve both North and South Indian food (this issue will be explored further in the next paragraph). Even if all strings are alphabetized, there were 1562 different cuisine types, and 80 different restaurant types due to there not being a limit on the number of cuisine types or restaurant types that each restaurant could be labeled as belonging to. This was especially problematic for a dataset of just over 7000 entries as there were many restaurants that were unique categories unto themselves using this form of coding. This would not be a fair test for one-hot encoding. Additionally, there was the issue of similarity.

The other challenge of one-hot encoding with so many unique categories is that it was not representative of the actual dataset. For example, while there were 80 different restaurant types, almost half of the restaurants had “quick bites” as one of the categories listed. This applied to many other cases in both restaurant and cuisine type (see appendix figures 4a and 4b). Thus, while one-hot encoding the columns without preprocessing would create a sparse space, most of the restaurants actually served similar foods and would probably be considered similar. Thus, I viewed the issue of one-hot encoding these columns as a text analysis problem.

The issue of one-hot encoding was solved by creating a document term matrix for the restaurant type and cuisine type features. With a proper corpus, the document term matrix approach would be equivalent to one-hot encoding by coding whether each restaurant had a specific cuisine listed and a specific restaurant type listed. To do this, I had to make the corpus only choose one word from any 2-word restaurant/cuisine types, and make sure no 2-word cuisine or restaurant type contained common words. Thus, I created a custom stop-word dictionary that included the words “Indian”,” “Food,” “Lankan,” and “dining” among others. When doing this, every restaurant and cuisine type was captured in the document term matrix, and with all cuisines reduced to one unique word, the final document term matrix was all [0,1] for every cuisine and restaurant type, thus resembling one hot encoding. This provided the benefit of reducing the feature space from over 1700 to 162 compared to a standard one-hot-encoding approach. It also made a similarity coefficient a realistic potential measure of distance for KNN instead a Minkowski metric such as Manhattan or Euclidean distance.

*Categorical Embedding*

For categorical embedding, the categorical\_embedder function was imported. Before running the embedder, I made sure that all string-lists from cuisine type and restaurant type were alphabetized so that restaurants of the same type were not miscategorized due to having different alphabetizations. The structure of the categorical embedder was a 2-layer ANN with 1000 ReLU neurons in the 1st layer and 500 ReLU neurons in the 2nd layer. The embedder is configured such that for each categorical variable, the number of embedding feature columns returned is limited to the minimum of half the number of factors in the feature and 50. After fitting the categorical variables through the embedder, the feature space contained 99 predictors (a reduction by 39%). This resulted in a proportional reduction of run-time for all the models under categorical embedding compared to the one-hot encoding.

*KNN*

For both one-hot and categorical embedding, a grid-search was used to determine the ideal k-parameter (from 1 to 100) for each distance metric of Euclidean, Manhattan, and cosine for both normal and distance weights. Using 5-fold cross-validation, the optimal parameters for the one-hot encoding model for binary prediction were k=47, normal weight, and cosine distance (suggesting that the document-term coding for one-hot was useful). For the 3-category prediction using one-hot encoding, the optimal parameters were the same except that the optimal k=65. When using categorical embedding, Euclidean distance became the optimal distance metric with normal weighting. For the 2-class problem, the optimal k was 29. For the 3-class problem, the optimal k was 82.

*Random Forest*

For Random Forest, the parameters checked using the grid search procedure were maximum tree depth and number of trees. For one-hot encoding for binary prediction, the optimal parameters were 450 estimators and tree depth of 10. For the 3-class problem, the optimal parameters were 400 estimators and tree depth of 40. When categorical embedding was used, the optimal parameters for binary classification were max depth of 10 and 250 estimators. For the 3-class problem, the optimal parameters were depth 10 and 450 estimators.

*ANN*

For ANN, the hyperparameters checked were number of hidden layers (1 or 2), neurons in each hidden layer (8 to 64 in the 1st hidden layer counting by 8, 0 to 16 neurons in the 2nd hidden layer counting by 2), and number of epochs (2 to 30 counting by 2). ReLU was the neuron type chosen. For one-hot encoding the optimal encoding for binary classification was 40 neurons in the 1st hidden layer and 4 neurons in the 2nd layer with 12 epochs as the optimal training time. For the 3-class classification with one-hot encoding the optimal parameters were 64 neurons in the 1st layer, 2 neurons in the 2nd layer, and 8 epochs training.

When categorical embedding was used, the optimal parameters for binary encoding were 64 neurons in the 1st layer, 4 neurons in the 2nd layer, and 18 epochs. For the 3-class task, the optimal configuration was 64 neurons in the 1st layer, 16 neurons in the 2nd layer, and 18 epochs. Because this was the upper limit of epochs, an additional search was done to see if more epochs performed better, but 18 remained the optimal number of epochs.

*Evaluation Metric*

As implied in the model tuning sections, accuracy was the metric by which models were compared and judged since both the binary and 3-category prediction targets were balanced classes. Using the mean and variance of the accuracies from the 5-fold cross validation findings, a t-test was conducted to test whether the accuracy differences were significant for each hypothesis. The confidence interval used was 95%. For the 3-category prediction, a confusion matrix was also generated to evaluate the error types.

**Results**

*One-hot encoding*

For both the binary and 3-class problem, Random Forest significantly outperformed both KNN and ANN (as can be seen in table 1). It had a 76.7% classification accuracy for the binary problem, and a 62.7% classification accuracy for the 3-class problem. When using the mean and variance of the accuracy scores from the 5-fold cross validation trials, the t-tests revealed a significant likelihood that Random Forest did in fact perform better than KNN and ANN. KNN especially struggled with one-hot encoding for the 3-class problem when compared to Random Forest as it had over 12% worse predictive accuracy. Thus, we accept the first hypothesis that Random Forest would significantly outperform both KNN and ANN for both the binary and 3-class prediction problem using one-hot encoding.



Table 1: Accuracy scores for one-got encoding and probability that KNN, Random Forest, and ANN are equivalent for both binary and 3-class problem.

When examining the confusion matrix for the model performances under one-hot encoding (table 2), it can be seen that KNN and ANN both over-predicted that a restaurant would be rated “bad” compared to Random Forest. ANN was able to predict both Good and Bad restaurants correctly but struggled to correctly predict scores in the “average” area (which were often misclassified as “bad”). Thus, Random Forest was overall a better choice for correctly predicting all 3 classes using one-hot encoding. However, ANN and KNN were narrowly better if one wanted to confidently avoid a bad restaurant. KNN was especially better at not misclassifying “bad” restaurants as “good.”



Table 2: Confusion matrices for 3-class Random Forrest, ANN and KNN predictions using one-hot encoding.

*Categorical Embedding*

Table 3 contains the binary and 3-class prediction accuracy for each technique, as well as the statistical tests for both the 2nd and 3rd research hypotheses. In comparing how categorical embedding changed the performance of all 3 models for both the binary and 3-class classification, KNN and ANN both saw upticks in predictive accuracy compared to categorical embedding. These gains in predictive accuracy were statistically significant for both the binary and 3-class problems. Random Forest, on the other hand, was not positively impacted by the use of categorical embedding. For binary prediction, it performed slightly worse on average (76.7% for one-hot encoding, 76.5% for categorical embedding), but this was not statistically significant, and the models should be considered equivalent. More surprising, the degradation of performance for the 3-class problem was considered statistically significant (62.7% for one-hot encoding vs. 59.1% for categorical embedding). This was likely due to the nature of the “cuisine type” and “restaurant type” categories since under the document-term encoding restaurants could belong to multiple categories while categorical embeddings could only belong to one possible embedding. This likely created additional sparsity in the data using a partitioning based classifier with so many unique combinations for so few data points. These results led me to accept the 2nd research hypothesis that categorical embedding would significantly improve the performance of KNN and ANN, while not significantly improving the performance of Random Forest.



Table 3: Accuracy for categorical embedding for KNN, ANN, and Random Forest, p-value that it is equal to one-hot encoding; and p-value that models are equivalent for binary and 3-class problem.

When comparing the 3 models to each other, the utility of categorical embedding was further revealed. KNN, which was the worst performing algorithm under one-hot encoding, had its performance raised to such that Random Forest was not significantly different (although the measured predictive accuracy remained higher for Random Forest for both binary and 3-class). The accuracy of ANN for 3-class prediction was also raised to the point where it statistically was no different than Random Forest (and they obtained virtually identical averages in this trial). Additionally, while not reflected in the table, I evaluated the p-value that ANN under categorical embedding performed equally to Random Forest under one-hot encoding since the results suggested that the one-hot embedding was better for Random Forest and there still was not a significant difference. For binary prediction, the results suggested that ANN significantly outperformed Random Forest with categorical embedding. Therefore, I accepted the 3rd research hypothesis that categorical embedding improved the accuracy of KNN and ANN to the point that they would either be statistically equivalent to Random Forest or would significantly outperform Random Forest for both binary and 3-class prediction.



Table 4: Confusion Matrix for 3-class Random Forrest, ANN and KNN predictions using categorical embedding.

In examining the confusion matrices for 3-class prediction with categorical embedding (Table 4), categorical embedding changed both the quality and nature of the performance of all 3 models. KNN saw an increase in correct predictions across all 3 classes, although it was still better at predicting “bad” restaurants due to over-predicting that a restaurant would be “bad.” Categorical embedding improved it to the point that it was by far the best at not misclassifying a “bad” restaurant as “good.” Random Forest under categorical embedding’s performance was more resembling of ANN and KNN in one-hot encoding in that it over-predicted “bad” and “good” restaurants. While it outperformed both ANN and KNN at correctly predicting the “good” and “bad” classes, it did have a higher rate of misclassifying “bad” restaurants as “good” than both KNN and ANN, and misclassifying “good” restaurants as “bad” compared to KNN. ANN overall was best at predicting the middle class and did the best job of balancing correctly predicting “good” and “bad” restaurants without misclassifying them as “bad and “good,” respectively. Thus, while the accuracy metric was undoubtfully raised for KNN and ANN by categorical embedding, one could make a case for any encoding/algorithm as more useful depending on the goals for prediction since each algorithm had different prediction patterns for the 3-class problem.

**Discussion**

Before discussing the implications of the findings there are some limitations that must be addressed. First, the dataset used had 2 non-traditional categorical features in “cuisine type” and “restaurant type.” Due to having so many unique combinations of types in each category for a dataset of just over 7000 entries, it made traditional one-hot encoding non-intuitive and impractical. While the document-term encoding seemed to work, it did add a layer of complexity to the experiment. Related to this, the categorical embedding was done on minimally edited categories. With more time, I would have experimented with categorically embedding the document-term matrix encoding. While this would have not resulted in the same dimensional reduction seen with the encoding that I did since the categories would have all been binary, the embedder may have been able to assign more meaningful values to this encoding than the 1652 unique categories present in restaurant type. This other encoding would also likely be more robust to handling new restaurants with never-before-seen combinations of cuisines/types than the one I used (assuming it does not contain any new categories).

Other limitations had to do with feature and model selection. I did not do any feature-selection optimization for this experiment primarily because the one-hot encoding challenge presented by “cuisine type” and “restaurant” type took so much time. While this experiment was more about comparing techniques under a control and experimental setting and the feature space was relatively low-dimensional before the encoding techniques expanded it, it must be stated that I cannot be sure that I used the optimal feature-space to evaluate the models under. Similarly, I had to put constraints on the range of hyperparameters tested due to time and computing limitations. Therefore, the conditions that the models were evaluated were under the best hyperparameters found in these ranges as opposed to an even more fully exhaustive search.

This experiment found that, as expected, Random Forest outperformed KNN and ANN on mixed-attribute features. However, the results after categorical embedding suggest that categorical embedding can be a useful tool for encoding mixed-attribute data not just for different types of Neural Networks, but also for other Machine Learning algorithms that struggle to parse mixed attribute features such as KNN. Both KNN and ANN had significant gains in performance in classification accuracy to the point that they were not considered significantly different from Random Forest after applying categorical embedding (and outperformed Random Forest in the case of ANN for binary prediction). This is significant since the confusion matrix showed that the models, while having similar accuracy performances, were stronger and weaker in different class predictions compared to each other. Thus, being able to use categorical embedding could allow researchers to build more reliable ensemble models using a mixture of Machine Learning techniques as opposed to relying on tree-based classifiers alone for mixed attribute data. Furthermore, even though Random Forest’s accuracy performance degraded under categorical embedding, this was likely due to the unique nature of this data, and it would not be expected with more conventional categorical representations of “cuisine type” and “restaurant type.”

The results of this experiment suggest more research should be conducted with more datasets on the utility of categorical embedding for mixed attribute data. A particular area needing scholarship is in how categorical embedding might help the predictive capabilities of other Machine Learning Algorithms that struggle with mixed-attribute data. The results categorical embedding could be a promising way to map categorical features to the continuous space in a way that benefits more Machine Learning algorithms than just ANN.

**Acknowledgements**

The author wishes to acknowledge the work of Shivand Roy in developing the categorical\_embedder package that was used. The documentation can be found at <https://pypi.org/project/categorical-embedder/>. The author also wants to thank his wife and daughter for understanding the late nights and time away getting this final project done.

**References**

Data Source- <https://www.kaggle.com/datasets/abhijitdahatonde/zomato-restaurants-dataset>

[1] S. Ahn, H. Ko, & J. Kang. “Regularized categorical embedding for effective demand forecasting of bike sharing system,” in *Studies in Computational Intelligence* vol. 929. 2021

[2] G. Tuerhong, M. Wishouer, & D. Zhang. “An improved k-nearest neighbor classifier for high-dimensional and mixture data.” *Journal of Physics: Conference Series.,* vol. 1813. 2020

[3] S. Bishnoi et al. “Classification of cotton genotypes with mixed continuous and categorical variables: Application of machine learning models,” *Sustainability.,* vol 14. 2022

[4] Y. Li et al. “CCAE: Cross-field categorical attributes embedding for cancer clinical endpoint prediction,” *Artificial Intelligence in Medicine.,* vol. 107. 2020

[5] T. Zhu. “Analysis on the Applicability of Random Forest.” *Journal of Physics: Conference Series.,* vol. 1607. 2020

[6] S. Luo et. al. “Non-numerical nearest neighbor classifiers with value-object hierarchical embedding.” *Expert Systems with Applications*. vol. 150. 2020

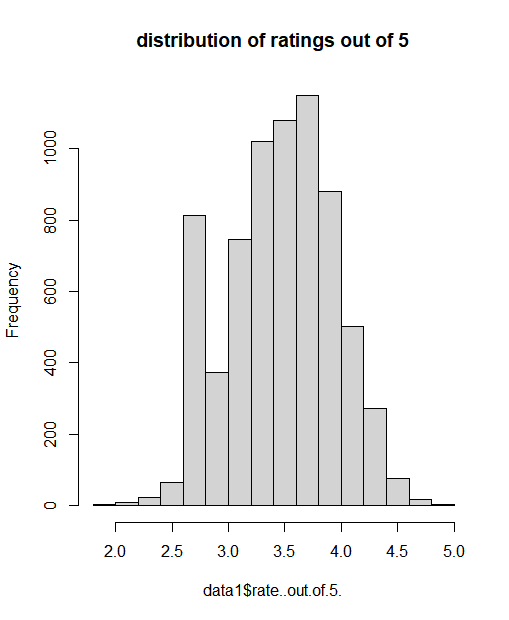
[7] M. Mueller, K. Gruber, & D. Folk. “Continuous Diffusion for Mixed-Type Tabular Data.” *NeurIPS Workshop on Synthetic Data Generation with Generative AI.* 2023

[8] M.K. Dahouda & I. Joe. “A deep-learned embedding technique for categorical features encoding,” *IEEE Access.* 2021

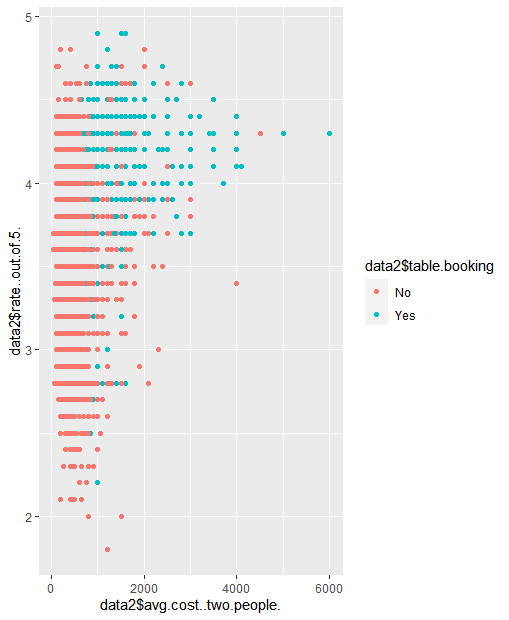
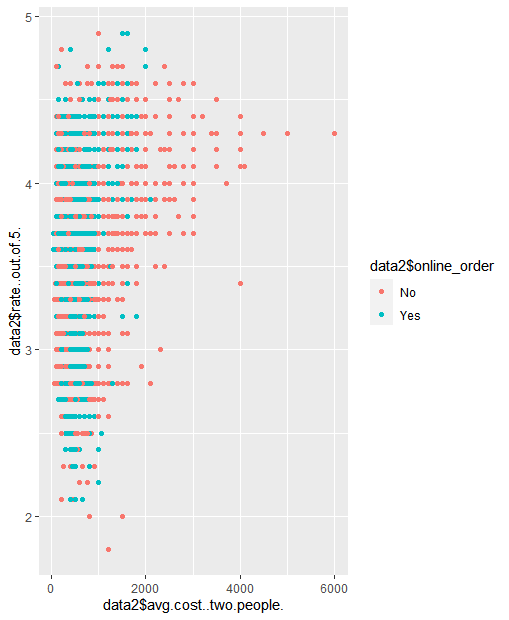
[9] Sarthank, S. Shukla, & S.P. Tripathi. “EmbPred30: Assessing 30-days readmission for diabetic patients using categorical embeddings,” in *Advances in Intelligent Systems and Computing,*. vol. 1168. 2020

[10] P. Shi & K. Shi. “Non-life insurance risk classification using categorical embedding,” *North American Actuarial Journal.,* vol 27(3),pp. 579-601, 2023

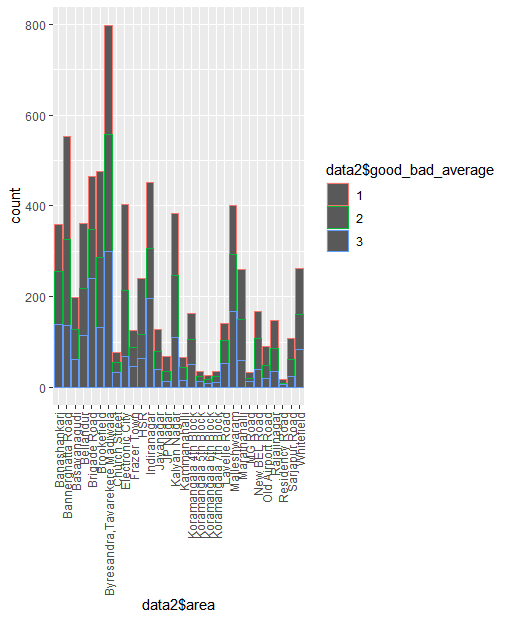
**Appendix**

****

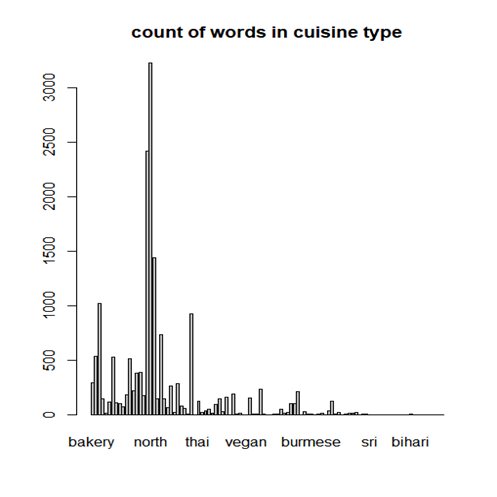
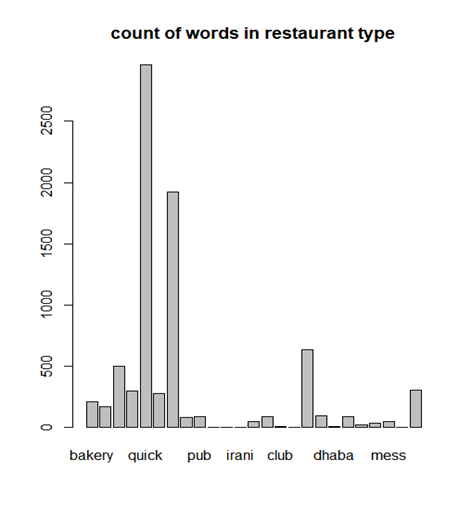
Appendix Figure 1: Distribution of Restaurant Ratings out of 5 in dataset

****

Appendix Figure 2a and 2b: Plot of rating out of 5 by average cost keyed by online ordering(2a) and table booking (2b).

****

Appendix Figure 3: Distribution of restaurants by area keyed by count of bad (1), average (2) and good (3) restaurants.



Appendix 4a and 4b: Word frequencies in document-term matrices for “restaurant type” (4a) and “cuisine type”(4b).