Encoding Methods for Categorical Data: A Comparative Analysis for Linear Models, Decision Trees, and Support Vector Machines

Andrei-Iustin Udilă[1]

Supervisor(s): Andra Ionescu[1]**, Asterios Katsifodimos**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Andrei-Iustin Udilă
Final project course: CSE3000 Research Project
Thesis committee: Andra Ionescu, Asterios Katsifodimos, Elvin Isufi

Abstract

This paper presents a comprehensive evaluation and comparison of encoding methods for categorical data in the context of machine learning. The study focuses on five popular encoding techniques: one-hot, ordinal, target, catboost, and count encoders. These methods are evaluated using linear models, decision trees, and support vector machines (SVMs).

The results demonstrate that one-hot encoding consistently achieves the highest accuracy across all evaluated machine learning algorithms. However, it also incurs a higher runtime, especially when feature cardinality is high. Catboost encoding emerges as a promising alternative, striking a balance between accuracy and runtime efficiency. The ordinal, target, and catboost encoders perform similarly, with small variations depending on the specific machine learning algorithm used.

Based on the findings, practitioners are advised to select one-hot encoding when accuracy is of utmost importance and computational resources are sufficient. For scenarios where runtime efficiency is critical, the catboost encoder offers competitive accuracy while minimizing training time. The ordinal encoder can be a suitable alternative when dealing with high feature cardinality.

1 Introduction

The transformation of categorical data into numerical data is an essential pre-processing step in many machine learning (ML) applications. Categorical data poses challenges when used in machine learning models, as they are often not directly comparable or measurable. To address this issue, a common approach is to convert categorical data into numerical data [1]. Various encoding techniques are available for this purpose, each with its own strengths and weaknesses depending on specific use cases [2].

Existing literature has explored different encoding techniques on various machine learning algorithms, but there is still much to be learned about which encoding technique is most effective for different types of models [2; 3; 4]. For example, one-hot encoding is more effective for tree-based algorithms [2], while target encoding works better for linear models and support vector machine (SVMs) [5]. This variation in results indicates that there is a need for further research on the comparative study of different encoding techniques on multiple machine learning algorithms.

The primary motivation for this research is to provide the reader with an understanding of when to choose a specific encoding method. To gain this knowledge, this paper analyses the following five encoders: One-Hot, Ordinal, Target, Catboost, and Count. Factors such as dataset properties, task (regression or classification), and ML model will also be taken into account.

Therefore, this paper will focus on the following research questions:

- What is the impact of encoding categorical data into numerical data on the performance (accuracy and runtime) of machine learning algorithms?
- Is there an encoding technique that is most effective for a certain ML algorithm?
- Can multiple encoding methods be combined to obtain better metrics?

This study's findings shed light on the efficacy of different encoding methods and offer insights into selecting the optimal technique for a given scenario. The main contribution of this research can be summarised as follows:

- Creating a baseline for evaluating the efficacy of different encoding techniques across a range of scenarios. This is carried out by encoding categorical data and subsequently training the models. Then, a comparison of the results for each method is performed.
- The use of multiple encoding methods on a single dataset is investigated. This involves defining a separate encoder for each categorical feature.
- The research determines whether the investment of resources to discover an appropriate encoding method for a specific use case is worthwhile compared to utilizing automated machine learning (AutoML) libraries, such as AutoGluon.
- Finally, the research investigates whether the results of this study can enhance existing data preprocessing pipeline designs. By examining the performance of current implementations, it is possible to assess whether investing resources in engineering a data encoding method when developing a new model is beneficial.

The findings expand upon the existing literature on data preprocessing, as presented in Section 2. Section 3 introduces the models and techniques used to compare encoding methods. Subsequently, the research methodology is described in Section 4. The experiments are summarized in Section 5, where the setup and results are presented. Moving forward, Section 6 concludes the paper, providing insights into the significance of the results and suggesting solutions for specific use cases. Section 7 offers an overview of the limitations of the research. Finally, in Section 8, the ethical implications of our findings are discussed.

2 Related Work

There has been research on methods for transforming categorical data into numerical data, since the data preprocessing step of encoding categorical variables plays a significant role in the quality of predictions [6]. Previous studies have briefly evaluated the performance of multiple methods on various machine learning algorithms, including decision trees, linear models, and neural networks. However, the lack of a thorough analysis that includes a wide variety of ML models, datasets and encoding techniques motivates this paper.

2.1 Encoding Methods on Neural Networks

A comparative study shows that target encoding and leave-one-out (LOO) encoding outperform other methods in terms

of classification accuracy, while binary encoding and count encoding are the most efficient in terms of training time [7]. Various techniques for encoding categorical variables in neural network classifiers were evaluated. The study examined six encoding methods, namely one-hot encoding, label encoding, binary encoding, count encoding, target encoding, and leave-one-out encoding, using multiple classification datasets. The authors compare the accuracy and training time of each encoding technique. The paper also discusses the advantages and disadvantages of each encoding method and provides insights on when to use each method based on the nature of the categorical variables and the classification task.

2.2 Comparative Analysis of Encoding Techniques

Encoders such as binary encoding, one-hot encoding, and feature hashing were analyzed in terms of predictive accuracy, model interpretability, and computational efficiency. Both simulated and real-world datasets were used to evaluate the performance of each encoding technique. The findings suggest that one-hot encoding and feature hashing perform better than binary encoding in terms of predictive accuracy and model interpretability, with feature hashing being the most computationally efficient technique [4].

Also, comprehensive surveys of data management techniques for machine learning have been performed, including data preprocessing techniques such as feature selection and data encoding. These surveys discuss the advantages and disadvantages of various encoding methods, such as one-hot encoding and label encoding, and highlight the importance of selecting an appropriate encoding method based on the characteristics of the data and the requirements of the ML algorithm [8; 9].

2.3 Feature selection and responsible data integration

Surveys have been conducted not only for comparing encoding techniques, but also for feature selection when working with categorical data. They also discuss the impact of feature selection on the performance of ML algorithms, highlighting the need for careful feature selection to avoid overfitting and improve model interpretability [9].

Lastly, the literature also focuses on the emerging challenges and ethical considerations in responsible data integration, such as data privacy, transparency, and fairness [10]. Frameworks have been developed for responsible data integration, incorporating these considerations to promote responsible data practices and mitigate potential harms.

In conclusion, despite extensive research on categorical data encoding methods, there is still a gap in understanding the optimal transformation technique for different machine learning algorithms. To address this gap, our research evaluates the effectiveness of multiple encoding methods on these specific machine learning algorithms. Through this evaluation, we aim to contribute to the development of a comprehensive understanding of the most suitable encoding techniques for different machine learning models.

3 Preliminaries

This section presents the models and techniques utilized in the research. Section 3.1 provides an overview of machine learning models, followed by the introduction of categorical data encoding methods in Section 3.2.

3.1 Machine Learning Models

Throughout this study, the machine learning models we use are linear models (linear and logistic regression), decision trees (XGBoost, LightGBM and random forests), and support vector machines. We chose these models for their simplicity and explainability. Also, these models offer different trade-offs and can help in making informed decisions when encoding categorical data for various machine learning tasks.

Linear models include logistic regression, linear discriminant analysis, and linear support vector machines, are a widely used class of machine learning algorithms. These models employ a linear function of the input features to predict the output variable [11].

Decision trees are used for classification and regression tasks. The model creates a tree-like structure where each internal node represents a decision based on a feature, and each leaf node represents a class or a value. Decision trees are popular due to their ability to handle non-linear relationships between features and target variables and their interpretability [11]. The decision tree algorithm has been improved over the years, with variations like random forests, boosting, and bagging algorithms that have been developed to improve its performance [12].

Random forests are a type of decision tree algorithm that builds a multitude of decision trees and combines them to get more accurate and stable predictions. Each tree is trained on a randomly sampled subset of the data and a subset of the features, which reduces overfitting and increases diversity among the trees.

Boosting is a family of algorithms that combine weak learners to create a strong learner. It works by iteratively training a sequence of weak models on re-weighted versions of the data, with each model attempting to correct the mistakes of its predecessor. The final prediction is a weighted combination of the predictions of all the models [13].

Support Vector Machines (SVMs) are a class of machine learning algorithms that are widely used for classification and regression tasks. The algorithm aims to find a hyperplane in a high-dimensional feature space that separates the data points into two or more classes, with the maximum margin between the classes. The maximum margin classifier is often referred to as the linear SVM. However, the algorithm can be extended to a non-linear form by using the kernel trick, allowing it to handle complex relationships between features and targets [11]. SVMs have been shown to perform well in many applications such as image classification, text categorization, and bioinformatics.

3.2 Encoding Methods

We specifically selected methods to ensure a broad coverage of both qualitative and quantitative approaches. Given the scale of this research, five of the most popular encoding methods were picked: one-hot, ordinal, target, catboost, and count.

One-Hot encoding involves creating a new binary feature for each category in the categorical variable. This technique is suitable for categorical variables with a small number of unique categories and is widely used in linear models, decision trees, and neural networks. The main drawback of one-hot encoding is that it can lead to a high-dimensional feature space, which can negatively impact the performance of some models and increase the computational cost [14]. The resulting feature matrix is sparse, where each column represents a category in the categorical variable, and each row represents a sample in the dataset.

Ordinal encoding assigns a unique integer value to each category in a categorical variable based on the order or rank of the categories, with lower integer values assigned to categories that are considered "lower" or "earlier" in the order. After encoding, we normalized the data, as some algorithms are sensitive to differences in scale between features [15]. One potential issue with ordinal encoding is that the integer values can introduce a numerical ordering to the categories that may not reflect the true relationships between the categories. This can be especially problematic for categorical variables with no inherent order or where the order is not well-defined.

Target encoding, also known as mean encoding, involves replacing each category with the mean (or some other aggregation) of the target variable for that category. The target variable is the variable we are trying to predict in our machine-learning model. The main advantage of target encoding is that it can provide a more informative representation of the data than one-hot encoding or ordinal encoding. By encoding the categories based on their relationship with the target variable, we can potentially capture more of the underlying structure in the data. However, target encoding can also introduce bias if the relationship between the categorical variable and the target variable is spurious or if there is overfitting. It is important to note that when using target encoding, it is essential to split the data into training and validation sets before encoding and computing the means using only the training set. Otherwise, there is a risk of data leakage, which can lead to overfitting and poor generalization performance.

Catboost encoding is similar to target encoding but also involves an ordering principle to overcome the problem of target leakage[16]. It uses a principle similar to the time series data validation. The values of the target statistic rely on the observed history, i.e. the target probability for the current feature is calculated only from the rows (observations) before it. With this approach, the first few observations in the dataset always have target statistics with much higher variance than the successive ones. To reduce this effect, many random permutations of the same data are used to calculate target statistics, and the final encoding is calculated by averaging across these permutations.

Count encoding, also known as frequency encoding, replaces each category in the categorical variable with its frequency or count in the dataset. This technique can be useful for variables with many categories, providing a more informative representation than One-Hot Encoding. However, it can introduce bias if the frequencies are not representative of the true underlying distributions of the categories. Additionally, it may not be suitable for variables with very few categories, as the frequency values may be too similar to one another.

4 Methodology

In the preliminaries chapter, we discussed the different encoding methods and machine learning algorithms used in this study. In the methodology chapter, we will provide a detailed outline of our experimental approach, including dataset selection, the data preprocessing steps and experimental design.

4.1 Hypotheses

Let us define two variables: **independent variable** as the encoding method and **dependent variable** as the evaluation metrics. The experiments are conducted by altering the independent variable, such as employing different encoding methods (one or multiple) for each dataset. This approach allows us to assess the performance of the machine learning models, our dependent variable. Performance is defined as a trade-off between accuracy/error and runtime. Given that we established what the variables in the experiments are, we can define hypotheses. For hypotheses that involve feature cardinality, a threshold of 5 has been set. Preliminary experiments have shown that when cardinality is higher than 5, the runtime grows exponentially in case of one-hot encoding. The hypotheses are:

- H1: One-Hot encoding performs better than the other encoders when the feature cardinality is lower than 5.
- H2: When cardinality is higher than 5, training a model using one-hot encoding takes significantly longer (more than 10%) than using other encoders.
- H3: Ordinal, Target or Catboost encodings should be selected instead One-Hot encoding when feature cardinality is higher than 5.
- H4: Combining encoding methods will improve performance.
- H5: The allocation of time and resources towards the design of an encoding method is justified compared to using solutions such as AutoML data preprocessing.

The research aims to validate the aforementioned hypotheses. Subsequently, the following subsection will outline the experimental methodology employed for this purpose.

4.2 Experiment Pipeline

The research is organized in a sequential manner, where the set of steps in the evaluation process can be envisioned as a pipeline, as in Figure 1.

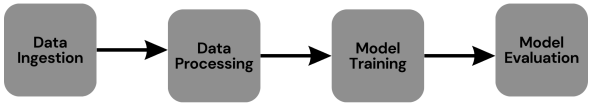


Figure 1: Experiment Pipeline

Dataset	Source	Task	#rows	#features(num., cat.)
Census Income	UCI	binary classification	48,842	14 (6, 8)
Bank Marketing	UCI	binary classification	45,211	17 (10, 7)
Nursery	UCI	binary classification	12,960	8 (0, 8)
Connect-4	UCI	multiclass classification	67,557	42 (0, 42)
Housing prices	Kaggle	regression	4,378	80 (36, 44)
Nasa Numeric	OpenML	regression	101	24 (4, 20)

Table 1: Overview of the datasets used in the evaluation

The experiment works as follows. A dataset is loaded. Then, a series of preprocessing steps are applied: first, we deal with missing values by applying simple imputation. This approach entails replacing missing values with the mean value of the available values. This method is frequently used in many studies due to its convenient nature as a reference technique [17]. Next, we apply the preprocessing step of encoding categorical values. This will be performed in five different ways, one for each encoding method. Again, this is our independent variable. Finally, the data is split into train and test sets, with 20% of the dataset to be used in testing.

To train the models, we first choose hyperparameters using grid search to ensure the quality of our results [18]. Then, the model is trained and finally, the resulting predictions on the test set are evaluated.

4.3 Datasets

We chose multiple datasets representing different types of data and classification tasks to evaluate the performance of encoding techniques in diverse scenarios. These publicly available datasets make it easy for other researchers to reproduce our experiments and compare their results.The datasets were gathered from Kaggle[1], OpenML[2], and UCI's machine learning repository[3].

Table 1 presents an overview of the selected datasets, including the dataset name, source, target variable, task, number of rows and features, and number of numerical and categorical features. We chose datasets with a mix of low to high numbers of total features, categorical features, and rows. Each type of instance was analyzed separately, and a comparison was made.

4.4 ML Models

Similarly to encoding methods, machine learning models from popular libraries were used for their reliability and efficiency. For linear models and decision trees, we used AutoGluon's TabularPredictor [4]. AutoGluon handles hyperparameter tuning. For SVMs, we used Sk-learn[5]'s implementation with grid search for choosing hyperparameters.

Linear Models. The linear models used are:

(i) Linear regression - for regression tasks.

(ii) Logistic regression - for classification tasks.

Decision Trees. The decision tree used (both for regression and classification) are:

(i) XGBoost - a distributed gradient-boosted decision tree.

(ii) LightGBM - a light gradient-boosting machine based on decision trees.

(iii) Random Forest - a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset

Support Vector Machines. The SVMs used are:

(i) SVR - for regression tasks.

(ii) SVC - for classification tasks.

The tuned hyperparameters for SVMs are C, gamma, kernel and degree (when the kernel was a polynomial) [19].

5 Evaluation

The section describes the evaluation setup and results. It also highlights the best-performing encoders and discusses the implications of combining encoding methods.

5.1 Metrics

We evaluated the performance of the encoding methods on classification and regression tasks. To accommodate these types of tasks, we chose the most suitable metrics per use case [20]. For classification tasks, the following metrics were used:

(i) **Accuracy**: percentage of correctly classified predictions on test data.

(ii) **Area under ROC curve**: the tradeoff between the true positive rate (TPR) and the false positive rate (FPR).

(iii) **F1-Score**: harmonic mean of precision and recall.

(iv) **Precision**: proportion of true positives among all predicted positives.

(v) **Recall**: proportion of true positives among all actual positives.

(vi) **Matthews Correlation Coefficient (MCC)**: takes into account true positives, true negatives, false positives, and false negatives to provide an overall measure of classification performance.

For regression tasks, the following metrics were used:

(i) **Mean Squared Error (MSE)**: the average squared difference between the predicted and actual values.

[1] www.kaggle.com

[2] www.openml.org

[3] https://archive.ics.uci.edu/ml/index.php

[4] https://auto.gluon.ai/0.4.2/api/autogluon.predictor.html

[5] https://scikit-learn.org/stable/modules/svm.html

feature cardinality	Encoder				
	one-hot	**ordinal**	**target**	**catboost**	**count**
0-2	✔				
3	✔				
4	✔	✔			
5-10		✔			
10-15		✔	✔		
15+		✔	✔		

feature characteristics	Encoder				
	one-hot	**ordinal**	**target**	**catboost**	**count**
no relationship	✔				
hierarchy		✔			
correlation to target			✔	✔	
rare features					✔

Table 2: Quantitative (left) and Qualitative (right) approaches of encoding categorical columns

(ii) **Root Mean Squared Error (RMSE)**: the square root of the MSE, which makes it more interpretable since it is in the same unit as the predicted values.

(iii) **Mean Absolute Error (MAE)**: the average magnitude of the differences between the actual and predicted values. It represents the average absolute deviation from the true values.

(iv) **R2 score (R2)**: measures the proportion of the variance in the dependent variable that is predictable from the independent variables.

Additionally, we measured the runtime, CPU usage, and memory usage of encoding the data, as well as training the ML model.

Overall, we evaluated the encoding methods based on their performance and efficiency, considering both the quality of the predictions and the computational resources required to obtain them.

5.2 Setup

This subsection details the evaluation setup such that the reader is able to reproduce the performed tests. The code is publicly available on a GitHub repository[6].

The experiments were performed using the datasets presented in Table 1. The code was run using Python 3.9, on an i5 9600K CPU and RTX2060 GPU. Missing data was handled using imputation, as described in Section 4.2.

Baseline Benchmark

To establish a performance baseline for the comparison, we employed AutoGluon's data preprocessing pipeline, which includes categorical data encoding. After the data is processed, the models are trained and evaluated. We do not perform any other data preprocessing.

Encoder Benchmark

The performance of five encoding methods will be compared. The process is similar for all encoders. Firstly, the categorical data is encoded using one of the methods. Then, the data is normalized to avoid differences in the scale of features. Next, five ML models are trained with the resulting dataset, as described in Section 4.4. Finally, the models are evaluated using the metrics in Section 5.1.

[6]https://github.com/delftdata/bsc_research_project_q4_2023

Combining Encoders

In addition to evaluating individual categorical data encoding methods, we also designed a test setup to assess the performance of combining multiple encoding methods on the same dataset. The objective of this setup is to explore the potential benefits of leveraging different encoding techniques in synergy to improve the predictive accuracy of machine learning models. The method used to define how to encode which column is the following. For every column in the dataset, each of the five encoders mentioned in Section 3.2 was applied, while employing one-hot encoding for the remaining columns. This process facilitated the variance of the encoding methods on a single column while keeping the encoding method constant for the other columns. This method resulted in two approaches to encoding the categorical data: a quantitative approach, based on the cardinality of the features, and a qualitative approach, based on the relationship between the features, both presented in Table 2. A tick illustrates that an encoder is suitable to use in the depicted case.

Notably, the one-hot encoder demonstrated superior performance when the feature cardinality was below 5. In cases where the cardinality ranged between 5 and 10, the ordinal encoder exhibited the best performance. Moreover, for categorical features with a cardinality exceeding 10, the target encoder emerged as the most effective encoding method in terms of performance.

Regarding the qualitative approach, we defined the 4 categories in Table 2. The one-hot encoder performed best when no relationship existed between the categories. The ordinal encoder was more effective for hierarchical relationships, such as education levels. The catboost and target encoders performed better when a correlation was present between the feature and the target value. In cases involving rare values, the count encoder outperformed other methods.

5.3 Results

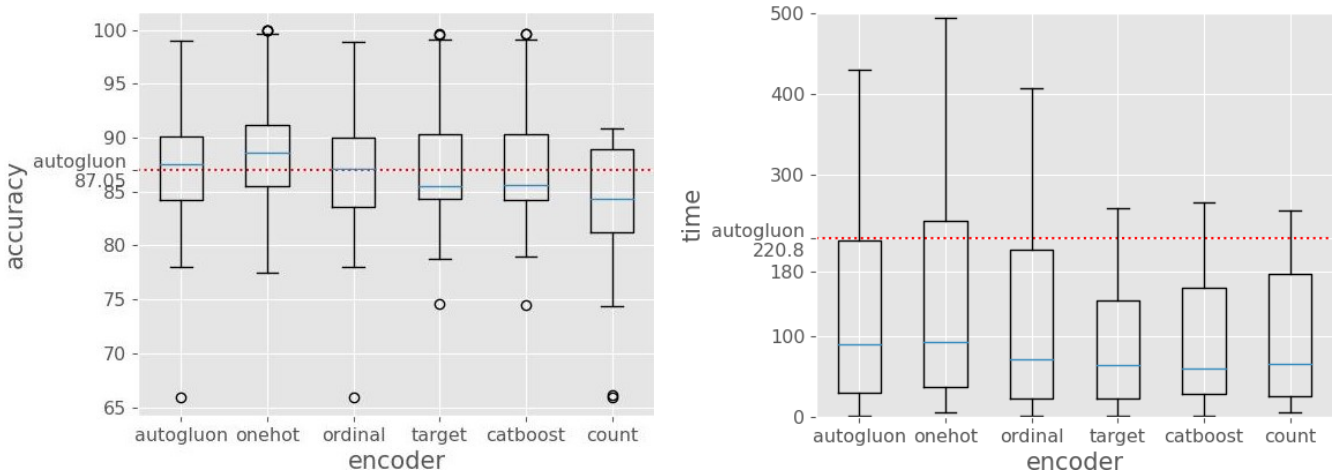
A summary of the results will be presented and analyzed in this section, while Appendix A contains full results.

One-Hot encoding performs best overall.
Depending on use case, other encoders should be used.

The five encoding methods were compared against AutoGluon as a baseline. The red dotted horizontal line in Figure 2 shows the performance of AutoGluon's automatic encoding.

Encoding Method	Linear		LightGBM		XGBoost		RandomForest		SVM	
	Accuracy	Runtime (s)	Accuracy	Runtime (s)	Accuracy	Runtime (s)	Accuracy	Runtime (s)	Accuracy	Runtime (s)
Onehot	**84.81**	**57**	85.91	39	**85.78**	**32**	84.24	5	79.88	33
Ordinal	80.78	23	**87.53**	**30**	85.28	30	83.98	4	79.88	10
Target	83.07	22	85.39	34	85.27	28	**84.76**	**5**	79.88	11
CatBoost	83.04	30	85.44	30	85.35	35	85.08	4	79.88	10
Count	79.02	22	85.24	35	85.32	29	84.36	6	**82.50**	**111**

Table 3: Evaluation results on Census Income dataset



(a) Prediction accuracy (%) of each encoder

(b) Runtime (s) of each encoder

Figure 2: Results summary on all classification datasets

The figures show boxplots of the accuracy and runtime of each encoder. The data was derived by computing the mean of the results across all ML algorithms analyzed. The dotted red line represents the results obtained when using AutoGluon to encode categorical data and serves as a baseline to compare to the encoders used by this study.

The y-axis represents the accuracy of the encoders tested in this experiment.

The results are consistent across all datasets. However, for simplicity, we will use the U.S. census income estimation dataset to present the results. A summary is visible in Table 3. On this specific dataset, the cardinality of the features is high, with only one feature having a cardinality lower than 5. As shown in Figure 2, one-hot encoding performed the best, outperforming both the other four encoders and the baseline. This proves hypothesis 1. At the same time, the runtime when using the one-hot encoder is 15% higher than using the ordinal encoder and up to 36% higher, compared to using the catboost encoder. This result proves hypothesis 2.

One should not be solely interested in finding the absolute best-performing encoder in terms of accuracy. When seeking a balance between runtime and performance, the most efficient approach is the implementation of catboost encoding. It achieves 0.62% accuracy per second spent training, while also being the second-best performer purely based on accuracy. On the opposite pole, the count encoder obtains 0.42% accuracy every second used for model training. Given that the other two encoding methods, namely ordinal and target, outperform one-hot encoding in this department, we can conclude that hypothesis 3 has been proven.

An interesting observation is that, in general, the ordinal, target, and catboost encoders perform similarly. Differences arise when comparing the results on individual ML algorithms. The results that we obtained are the following:

(i) **Linear Models:** One-Hot encoding performs best. However, when feature cardinality is high, target and catboost encoders are a better option as the accuracy drop is low compared with the gain in saved runtime.

(ii) **Decision Trees:** The best-performing encoder is also one-hot. However, the ordinal encoder performs between 0 and 1% worse, with a significantly lower runtime.

(iii) **SVMs:** One-Hot encoding has the highest accuracy, but that comes with the downside of a 3-times higher runtime. When considering the runtime, catboost encoding performs best, with target encoding being a close runner-up. This result was specifically expected since the increase of data dimensionality influences the runtime on SVMs the most.

Another observation is that in the context of the *Nursery* dataset, the utilization of the Count Encoder proved ineffective due to the presence of an equal number of instances for each category. Consequently, the resulting encoded columns exhibited identical values, thereby diminishing the utility of this particular encoding technique.

In general, using an implementation of an encoding method performs better than relying on AutoML alternatives, with performance gaps of at most 3% in accuracy compared to the best alternative.

Combining encoders does not improve performance. Combining multiple encoding methods did not lead to improved results. The accuracy obtained was lower for classification and the root mean squared error was higher for regression tasks when compared to using a single encoder on the whole dataset. This result disproves hypothesis 4. We identified three main problems with this approach:

(i) **Inconsistency**: Inconsistent encoding schemes can introduce noise or confusion for the models. This is mainly visible for linear models and SVMs.

(ii) **Information loss:** Different encoding methods capture and represent categorical information differently. By applying different encoders to different columns, information loss might be inadvertently introduced or the original relationships between features may be distorted.

(iii) **Correlations and interactions:** Combining encoding methods across columns can make it challenging for the model to identify correlations or interactions between the encoded features. Some models, such as decision trees, can struggle to recognize patterns effectively if the encoding schemes vary across features.

This result was consistent for both the quantitative and qualitative approaches to defining encoders for each column. Figure 3 shows the results for each ML model tested.

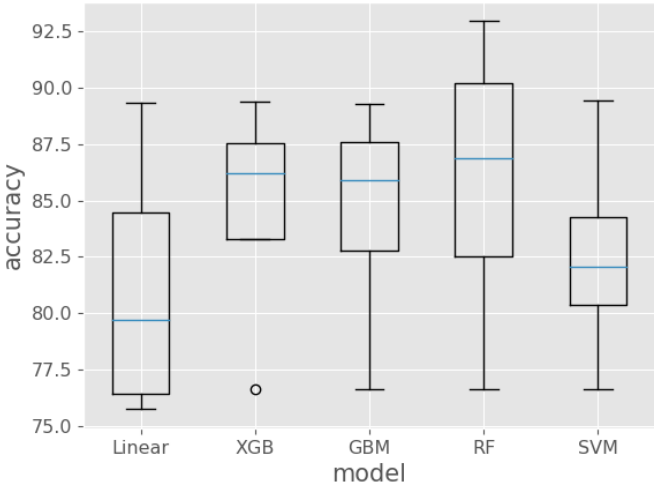


Figure 3: Results summary on combining encoders

6 Discussion and Conclusion

The results of the evaluation shed light on the performance and efficiency of different encoding methods for categorical data, providing valuable insights for selecting an appropriate encoder. In this subsection, we will provide a personal perspective and opinion on the results, while addressing the hypotheses set forth in this study.

Hypothesis 1 stated that one-hot encoding would outperform the other encoders. The results confirmed this hypothesis, demonstrating that one-hot encoding achieved the highest accuracy across linear models, decision trees and SVMs. Therefore, when accuracy is of paramount importance and the computational resources are sufficient, one-hot encoding should be the encoder of choice.

Hypothesis 2 suggested that one-hot encoding would have higher runtime compared to other encoders. The results supported this hypothesis, revealing that one-hot encoding incurred a significant runtime cost, particularly when the feature cardinality was high. This was clearly visible in the case of SVMs. This trade-off between accuracy and runtime highlights the importance of considering computational constraints and time limitations. In scenarios where runtime efficiency is a priority and a slight decrease in accuracy is acceptable, the ordinal or catboost encoders would be more suitable choices.

Hypothesis 3 posited that catboost encoding would provide the most efficient approach in terms of both accuracy and runtime. The results partially supported this hypothesis, as catboost encoding exhibited competitive accuracy levels while requiring less training time compared to one-hot encoding. This result was supported mostly in the case of SVMs. However, it did not consistently outperform other encoders in terms of accuracy. Nevertheless, catboost encoding emerges as a strong contender, striking a balance between performance and efficiency, making it an appealing choice in situations where runtime is a critical factor.

Hypothesis 4 examined the possibility of combining multiple encoding methods for improved results. However, the results disproved this hypothesis, revealing that combining encoders led to lower classification accuracy and higher regression root mean squared error. This finding emphasizes the importance of consistency and preserving the original relationships between features. It also underscores the challenges associated with incorporating diverse encoding schemes, which can hinder the model's ability to recognize patterns effectively, especially for decision trees. Thus, it is advisable to use a single encoding method across the entire dataset to avoid introducing noise and confusion.

Hypothesis 5 suggests that investing time and resources in designing an encoding method is justified compared to relying on solutions such as AutoML for data preprocessing. The results consistently showed that using an implementation of an encoding method outperformed the AutoML alternative. This finding supports hypothesis 5 and highlights the importance of carefully designing and selecting encoding methods tailored to the specific dataset and machine learning task at hand. While AutoML solutions offer convenience and automation, they may not always capture the nuances of the data and the underlying relationships between features as effectively as custom encoding methods.

Conclusion. Considering the trade-offs between accuracy and runtime, as well as the challenges associated with combining encoders, our recommendation would be to use one-hot encoding when the priority is maximizing accuracy and computational resources are not a limiting factor. However, in real-world scenarios where runtime efficiency is critical,

the catboost encoder stands out as a pragmatic choice, offering competitive accuracy while minimizing training time. The ordinal encoder can be a suitable alternative when feature cardinality is high, as it demonstrates comparable accuracy to one-hot encoding but with a significantly lower runtime.

It is important to acknowledge the limitations of this study. The findings are not universally applicable and depend on the specific dataset characteristics, feature distributions, and modelling objectives. The choice of encoding method should be made based on these considerations and the available computational resources. Future research should explore advanced encoding techniques, ensemble strategies, and evaluate the performance of encoding methods on larger and more diverse datasets.

In conclusion, the results provide valuable guidance for selecting encoding methods for categorical data. By considering the trade-offs between accuracy and runtime, researchers and practitioners can make informed decisions on which encoder to use based on their specific requirements and constraints.

7 Limitations and Future work

There are several avenues for further exploration and improvement in the field of encoding methods for categorical data:

Investigate advanced encoding techniques. This study focused on widely-used encoding methods, but there are emerging techniques that could be explored further. Techniques such as entity embedding, target encoding with smoothing, and other novel approaches may offer improved performance or efficiency.

Explore ensemble encoding strategies. Instead of using a single encoding method, ensemble techniques that combine multiple encoders or dynamically select the most suitable encoder for each feature could be investigated. Machine learning could be employed to do this [21]. This could potentially address the challenges associated with combining encoders observed in this study.

Evaluate on larger and diverse datasets. The evaluation of encoding methods can be extended to larger and more diverse datasets to assess their generalizability. Additionally, considering different application domains and datasets with varying characteristics could provide deeper insights into the strengths and weaknesses of different encoding methods.

Consider other machine learning algorithms. This study focused on linear models, decision trees, and SVMs. Evaluating the performance of encoding methods on other algorithms, such as neural networks, ensemble models, or gradient boosting machines, could provide a more comprehensive understanding of their effectiveness across a broader range of models.

Conduct a comprehensive analysis of computational costs. While this study considered runtime, cpu and memory usage as a measure of efficiency, a more comprehensive analysis of computational costs, including scalability, could provide a more nuanced understanding of the trade-offs associated with different encoding methods.

By addressing these future research directions, we can continue to enhance the understanding and utilization of encoding methods for categorical data, ultimately leading to more accurate and efficient machine learning models in real-world applications.

8 Responsible Research

Ethical Considerations

We acknowledge that our research involves working with sensitive personal data, as the datasets used in our experiments may contain features linked to individuals. To ensure privacy and confidentiality we took several measures. Firstly, we obtained datasets from reliable sources that had already anonymized the data (OpenML, UCI Repository, Kaggle). Secondly, we only used the data for research purposes and did not share it with third parties. Lastly, we followed standard data protection regulations and guidelines while working with the data. We believe that our research meets ethical standards for conducting research with sensitive personal data.

Limitations

The limitations of the research were explicitly addressed in the corresponding section. These limitations include the specific encoding methods and machine learning algorithms considered and the characteristics of the datasets used. It is essential for future research to expand upon these limitations by exploring additional encoding methods, diverse datasets, and real-world applications. By acknowledging these limitations, the research encourages further investigation and improvement in the field

Reproducibility

To ensure the reproducibility of our methods, we provide detailed descriptions of our methodology and experimental work sections. We have also made our code and dataset publicly available on a code-sharing platform. In addition, we have used standard machine learning algorithms and evaluation metrics that are commonly used in the literature, which enhances the reproducibility of our results.

9 Acknowledgements

We would like to express our heartfelt appreciation to our supervisor, Andra Ionescu, for her guidance, expertise, and continuous support throughout the entire research process. Her insightful feedback and constructive suggestions greatly contributed to the success of this study.

We would also like to express our gratitude to Professor Asterios Katsifodimos for his invaluable mentorship and supervision. His expertise and guidance have been instrumental in shaping the direction of this research project.

Lastly, we would like to acknowledge the contributions of the peers who participated in this study by providing their time and valuable insights. Their involvement has been vital in gathering the necessary data and ensuring the completion of this research.

We are truly grateful for the support and encouragement we have received from all those involved, without which this study would not have been possible.

References

[1] Canchen Li. Preprocessing methods and pipelines of data mining: An overview. In *Seminar Data Mining*. Technical University of Munich, 2019.

[2] Diogo Seca and João Mendes-Moreira. Benchmark of encoders of nominal features for regression. In *Trends and Applications in Information Systems and Technologies*, pages 146–155. Springer International Publishing, 2021.

[3] Florian Pargent, Bernd Bischl, and Janek Thomas. *A benchmark experiment on how to encode categorical features in predictive modeling*. PhD thesis, Master Thesis in Statistics, Ludwig-Maximilians-Universität München, 2019.

[4] Cedric Seger. An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing. Bachelor's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2018.

[5] Florian Pargent, Florian Pfisterer, Janek Thomas, and Bernd Bischl. Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. *Computational Statistics*, 37(5):2671–2692, 2022.

[6] John Hancock and Taghi Khoshgoftaar. Survey on categorical data for neural networks. *Journal of Big Data*, 7, 04 2020.

[7] Kedar Potdar, Taher Pardawala, and Chinmay Pai. A comparative study of categorical variable encoding techniques for neural network classifiers. *International Journal of Computer Applications*, 175:7–9, 2017.

[8] Chengliang Chai, Jiayi Wang, Yuyu Luo, Zeping Niu, and Guoliang Li. Data management for machine learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[9] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–45, 2017.

[10] Fatemeh Nargesian, Abolfazl Asudeh, and HV Jagadish. Responsible data integration: Next-generation challenges. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 2458–2464, 2022.

[11] Zhi-Hua Zhou. *Machine Learning*, pages 58, 80, 130. Springer Singapore, 2022.

[12] Carl Kingsford and Steven L. Salzberg. What are decision trees? *Nature Biotechnology*, 26(9):1011–1013, 2008.

[13] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[14] Ivan Lopez-Arevalo, Edwin Aldana-Bobadilla, Alejandro Molina-Villegas, Hiram Galeana-Zapién, Victor Muñiz-Sanchez, and Saul Gausin-Valle. A memory-efficient encoding method for processing mixed-type data on machine learning. *Entropy*, 22(12), 2020.

[15] Dalwinder Singh and Birmohan Singh. Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, 97:105524, 2020.

[16] John T Hancock and Taghi M Khoshgoftaar. Catboost for big data: an interdisciplinary review. *Journal of big data*, 7(1):1–45, 2020.

[17] Tlamelo Emmanuel, Thabiso Maupong, Dimane Mpoeleng, Thabo Semong, Banyatsang Mphago, and Oteng Tabona. A survey on missing data in machine learning. *Journal of Big Data*, 8(1):140, 2021.

[18] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1):1934–1965, 2019.

[19] Chih-wei Hsu, Chih-chung Chang, and Chih-Jen Lin. A practical guide to support vector classification chih-wei hsu, chih-chung chang, and chih-jen lin. *National Taiwan University*, 2003.

[20] Kathrin Blagec, Georg Dorffner, Milad Moradi, and Matthias Samwald. A critical analysis of metrics used for measuring progress in artificial intelligence. *University of Vienna*, 2021.

[21] Mwamba Kasongo Dahouda and Inwhee Joe. A deep-learned embedding technique for categorical features encoding. *IEEE Access*, 9:114381–114391, 2021.

A Detailed experiment results

A.1 U.S. Census Income Estimation

Encoder	Model	Metrics					
		Accuracy	AUC	F1	Precision	Recall	Time(s)
AutoML	Linear	84.55	0.90	0.63	0.70	0.57	30
	XGB	87.55	0.92	0.70	0.78	0.64	30
	GBM	87.62	0.92	0.71	0.78	0.65	35
	RF	86.11	0.91	0.67	0.74	0.62	3
	SVM	79.88	0.58	0.30	0.80	0.18	10
Onehot	Linear	84.81	0.90	0.64	0.71	0.58	57
	XGB	85.78	0.91	0.66	0.74	0.60	32
	GBM	85.91	0.91	0.67	0.73	0.61	39
	RF	84.24	0.89	0.63	0.69	0.57	5
	SVM	79.88	0.58	0.30	0.80	0.18	33
Ordinal	Linear	80.78	0.82	0.43	0.69	0.31	23
	XGB	85.28	0.90	0.64	0.74	0.58	30
	GBM	87.53	0.93	0.70	0.78	0.64	30
	RF	83.98	0.89	0.61	0.70	0.54	4
	SVM	79.88	0.58	0.30	0.80	0.18	10
Target	Linear	83.07	0.86	0.57	0.69	0.49	22
	XGB	85.27	0.90	0.64	0.74	0.56	28
	GBM	85.39	0.90	0.64	0.64	0.57	34
	RF	84.76	0.89	0.61	0.74	0.52	5
	SVM	79.88	0.58	0.30	0.80	0.18	11
CatBoost	Linear	83.04	0.90	0.63	0.70	0.57	30
	XGB	85.35	0.90	0.64	0.74	0.57	35
	GBM	85.44	0.90	0.65	0.74	0.58	30
	RF	85.08	0.89	0.62	0.75	0.54	4
	SVM	79.88	0.58	0.30	0.80	0.18	10
Count	Linear	79.92	0.80	0.42	0.64	0.31	22
	XGB	85.32	0.90	0.64	0.73	0.56	29
	GBM	85.24	0.90	0.64	0.74	0.56	35
	RF	84.36	0.89	0.62	0.71	0.55	6
	SVM	82.50	0.68	0.53	0.70	0.42	111

Table 4: Summary of results for U.S. Census Income Dataset Results

A.2 Bank Marketing

Encoder	Model	Metrics					
		accuracy	AUC	F1	Precision	Recall	Time(s)
AutoML	Linear	88.99	0.86	0.29	0.57	0.20	34
	XGB	90.27	0.90	0.46	0.64	0.36	70
	GBM	90.12	0.90	0.43	0.64	0.32	110
	RF	90.24	0.90	0.45	0.64	0.35	123
	SVM	88.39	0.5	0.94	0.88	1.0	19
Onehot	Linear	89.77	89.69	41.71	0.61	0.31	53
	XGB	90.77	0.92	0.51	0.67	0.41	74
	GBM	90.51	0.92	0.49	0.65	0.4	111
	RF	90.41	0.92	0.47	0.65	0.37	122
	SVM	88.39	0.5	0.94	0.88	1.0	28
Ordinal	Linear	88.99	0.86	0.3	0.57	0.2	23
	XGB	90.31	0.9	0.46	0.65	0.36	54
	GBM	90.12	0.9	0.43	0.65	0.33	89
	RF	89.97	0.9	0.45	0.62	0.35	100
	SVM	88.39	0.5	0.94	0.88	1.0	19
Target	Linear	89.33	0.88	0.36	0.59	0.26	24
	XGB	90.36	0.91	0.46	0.66	0.35	55
	GBM	90.38	0.91	0.47	0.65	0.37	90
	RF	90.3	0.91	0.48	0.64	0.39	101
	SVM	88.39	0.5	0.94	0.88	1.0	20
CatBoost	Linear	89.34	0.88	0.36	0.59	0.26	23
	XGB	90.41	0.91	0.47	0.66	0.36	53
	GBM	90.35	0.91	0.45	0.66	0.34	88
	RF	90.31	0.91	0.48	0.64	0.39	99
	SVM	88.39	0.5	0.94	0.88	1.0	20
Count	Linear	89.52	0.88	0.4	0.6	0.3	23
	XGB	90.66	0.93	0.52	0.64	0.44	54
	GBM	90.88	0.93	0.54	0.65	0.45	90
	RF	90.56	0.92	0.52	0.63	0.44	103
	SVM	88.39	0.5	0.94	0.88	1.0	20

Table 5: Bank Marketing Dataset Results

A.3 Nasa Numeric

Encoder	Model	Metrics				
		RMSE	MSE	MAE	R2	Runtime(s)
AutoML	Linear	635	404089	-457.3	-0.46	22
	XGB	536	287755	-324.7	-0.04	57
	GBM	517	268010	-402.91	0.03	74
	RF	614	377247	-385.88	-0.37	75
	SVM	427	182450	273.61	0.34	0.1
Onehot	Linear	1939	3761158	-1459.4	-12.62	14
	XGB	377	142633	-234.22	0.48	34
	GBM	486	236310	-382.8	0.14	516
	RF	621	386877	-403.77	-0.4	52
	SVM	427	182587	273.28	0.34	0
Ordinal	Linear	681	464400	-479.74	-0.68	13
	XGB	490	240237	-306.84	0.13	34
	GBM	533	284167	-409.13	-0.03	51
	RF	600	360350	-380.06	-0.3	51
	SVM	427	182450	273.6	0.34	0.1
Target	Linear	767	588880	-526.14	-1.13	13
	XGB	521	271774	-296.86	0.02	32
	GBM	568	323342	-384.64	-0.17	53
	RF	482	232650	-309.29	0.16	53
	SVM	409				0.1
CatBoost	Linear	650	423651	-531.71	-0.53	13
	XGB	452	204408	-283.79	0.26	33
	GBM	581	337999	-414.74	-0.22	52
	RF	403	162941	-272.87	0.41	53
	SVM	428	183650	308.84	0.34	0.1
Count	Linear	1091	1192006	-734.23	-3.31	13
	XGB	410	168920	-254.76	0.39	32
	GBM	683	466555	-440.17	-0.69	56
	RF	435	190071	-272.86	0.31	57
	SVM	407	166259	243.53	0.4	0.1

Table 6: Nasa Numeric Dataset Results

A.4 Nursery

Encoder	Model	Metrics		
		accuracy	Matth. Corr. Coef.	Runtime(s)
AutoML	Linear	78.01	0.68	53
	XGB	98.96	0.98	162
	GBM	98.88	0.98	309
	RF	97.49	0.96	314
	SVM	89.0	0.84	1
Onehot	Linear	92.4	0.89	73
	XGB	99.96	1.0	157
	GBM	100.0	1.0	272
	RF	99.61	0.99	275
	SVM	100.0	1.0	6
Ordinal	Linear	78.01	0.68	51
	XGB	98.88	0.98	159
	GBM	98.88	0.98	307
	RF	97.3	0.96	313
	SVM	89.0	0.84	1
Target	Linear	85.69	0.79	48
	XGB	99.5	0.99	137
	GBM	99.58	0.99	258
	RF	99.07	0.99	263
	SVM	86.5	0.8	1
CatBoost	Linear	85.69	0.79	47
	XGB	99.61	0.99	138
	GBM	99.65	0.99	261
	RF	99.07	0.99	266
	SVM	86.5	0.8	1

Table 7: Nursery Dataset Results (multiclass)

A.5 Housing Prices

Encoder	Model	Metrics				
		RMSE	MSE	MAE	R2	Runtime(s)
AutoML	Linear	40004	1600369404	-24040.09	0.78	48
	XGB	35148	1235448683	-20127.86	0.83	186
	GBM	38537	1485149547	-20623.46	0.79	297
	RF	42217	1782353377	-23399.35	0.75	306
	SVM	73846	5453298867	47058.24	0.24	1
Onehot	Linear	32874	1080751629	-19753.14	0.85	50
	XGB	33344	1111853870	-19183.52	0.84	152
	GBM	35850	1285246298	-19576.9	0.82	277
	RF	36848	1357803130	-21179.99	0.81	289
	SVM	76183	5803912840	49096.77	0.19	1
Ordinal	Linear	40229	1618435088	-23946.43	0.77	44
	XGB	35439	1255958853	-20450.94	0.82	122
	GBM	38008	1444610351	-20584.67	0.8	347
	RF	43009	1849819925	-23816.99	0.74	356
	SVM	73846	5453359326	47058.5	0.24	1
Target	Linear	32691	1068733614	-19901.42	0.85	44
	XGB	33705	1136048000	-16789.13	0.84	114
	GBM	32958	1086254201	-16978.21	0.85	201
	RF	35556	1264289761	-18260.46	0.82	209
	SVM	84246	7097495749	55638.82	0.0	1
CatBoost	LR	36600	1339592091	-21005.31	0.81	42
	XGB	33377	1114078686	-16824.02	0.84	144
	GBM	32631	1064789991	-16727.33	0.85	229
	RF	33904	1149534789	-18090.82	0.84	237
	SVM	54970	3021725343	30769.19	0.58	0.1
Count	Linear	36500	1332256859	-22641.89	0.81	44
	XGB	35030	1227123673	-20137.92	0.83	121
	GBM	37087	1375488953	-20324.23	0.81	249
	RF	42762	1828628600	-23890.4	0.74	258
	SVM	40179	1614426300	22357.49	0.77	0.1

Table 8: Housing Prices Dataset Results (regression)

A.6 Connect-4

Encoder	Model	Metrics		
		accuracy	Matth. Corr. Coeff.	Runtime(s)
AutoML	Linear	65.97	0.1	188
	XGB	86.55	0.72	429
	GBM	86.43	0.72	1145
	RF	82.9	0.64	1196
	SVM	83.05	0.6	155
Onehot	Linear	75.71	0.47	233
	XGB	86.89	0.73	494
	GBM	87.06	0.73	1132
	RF	84.51	0.68	1189
	SVM	88.85	0.73	176
Ordinal	Linear	65.98	0.1	174
	XGB	86.55	0.72	406
	GBM	86.67	0.72	1083
	RF	82.9	0.64	1132
	SVM	83.85	0.61	153
Target	Linear	74.58	0.44	72
	XGB	84.78	0.68	250
	GBM	84.72	0.68	831
	RF	78.72	0.54	905
	SVM	80.78	0.58	121
CatBoost	Linear	74.5	0.43	68
	XGB	84.65	0.68	228
	GBM	84.57	0.68	635
	RF	78.95	0.55	703
	SVM	80.78	0.58	133
Count	Linear	65.92	0.06	66
	XGB	82.53	0.63	256
	GBM	82.82	0.64	799
	RF	74.4	0.44	877
	SVM	66.12	0.0	242

Table 9: Connect-4 Dataset Results (multiclass)