



University of St.Gallen

University of St. Gallen School of Management, Economics,
Law, Social Sciences and International Affairs (HSG)

Dealing with Categorical Predictors to mitigate Information Loss in Supervised Machine Learning

Master in Quantitative Economics and Finance

August 23, 2021

Author: **Daniel Brunner**

Supervisor: **Prof. Ph.D. Michael Knaus**

Co-Supervisor: **Prof. Dr. Jana Mareckova**

Abstract

This thesis presents a collection of approaches to deal with categorical variables in a regression context that do not waste group information, as happens by using one-hot encoding. Among these are encoding methods such as low rank encoding, entity embedding, and regularized target encoding, as well as lasso-related algorithms like the group lasso, sparse-group lasso, and fused lasso. Furthermore, a categorical super learner is built from these methods. In a simulation study and on real data with focus on high-cardinality categorical variables, the predictive accuracy of the methods is evaluated. The simulation shows that the performance of each approach improves by increasing the number of levels of the categorical predictors. However, with real data only regularized target encoding combined with the OLS as well as the fused lasso could deliver stable results.

Keywords: categorical variables; preserve group information; encoding techniques; Machine Learning algorithms; regression; prediction accuracy

Contents

Abstract	I
List of Figures.....	IV
List of Tables	V
1 Introduction	1
2 Related work	3
3 Theoretical Background.....	6
3.1 Supervised Machine Learning	6
3.2 Categorical Variables.....	7
3.2.1 Characteristics of Categorical Variables	7
3.2.2 Encoding of Categorical Predictors	8
3.2.3 Pitfalls of One-Hot Encoding.....	8
3.2.4 Solution to the Dimensionality Problem of One-Hot Encoding.....	9
4 Dealing with Categorical Variables	11
4.1 Encoding Procedures preserving Group Membership	11
4.1.1 Low Rank Encoding	11
4.1.2 Entity Embedding.....	15
4.1.3 Regularized Target Encoding.....	18
4.2 Regression methods with Group Penalization	20
4.2.1 Group Lasso.....	20
4.2.2 Sparse-Group Lasso	22
4.2.3 Fused Lasso.....	23
5 Super Learner.....	24
5.1 Benefits of a Super Learner.....	24
5.2 Stacked Generalization	24
5.3 Including Algorithms into a Super Learner	26

6 Experiments.....	27
6.1 Simulations	27
6.1.1 Data Generating Process	27
6.1.2 Simulation Setup	28
6.2 Real Data	30
6.3 Implementation of Methods	31
6.3.1 Low Rank Encoding	31
6.3.2 Regularized Target Encoding	32
6.3.3 Entity Embedding	32
6.3.4 Group Lasso	32
6.3.5 Sparse-Group Lasso	33
6.3.6 Fused Lasso	33
6.4 Results	33
6.4.1 Simulation	34
6.4.2 Real Data	35
6.5 Discussion	35
7 Conclusion	39
References.....	VI
Appendix	IX
Appendix A: R-Code	IX
Appendix B: Simulations in Detail	X
Declaration of authorship.....	XI

List of Figures

Figure 1: Example of label encoding and one-hot encoding	8
Figure 2: Illustration of the sufficient latent state assumption	12
Figure 3: Intuition behind the group-wise means	13
Figure 4: Application scheme for low rank encoding	14
Figure 5: A basic neural network architecture with embeddings	16
Figure 6: Implementation example of entity embedding	17
Figure 7: Relative performance of the methods	38

List of Tables

Table 1: Algorithm of low rank encoding	14
Table 2: Algorithm of GLMM encoding regression	20
Table 3: Algorithm of a super learner	25
Table 4: Data-generating process for the simulations	28
Table 5: Overview of the simulation study	30
Table 6: Description of real-world datasets	31
Table 7: Results of the simulations	34
Table 8: Prediction accuracy on real datasets	35

1 Introduction

Many data analysts as well as researchers often use one-hot encoding when encountered with categorical variables in regression or classification problems (Johannemann, Hadad, Athey & Wager, 2019, p. 1). As most machine learning algorithms require numerical inputs to run, this is not a poor strategy to employ. However, despite offering certain advantages, one-hot encoding also comes with some shortcomings that become particularly severe when dealing with high-cardinality variables (Guo & Berkahn, 2016, p. 1). First, the data becomes sparse, and second, group membership is lost because the newly created dummy variables are independent variables in the regression model (Guo & Berkahn, 2016, p. 1). Therefore, there are some approaches that make use of group information and thus achieve an advantage, especially with high-cardinality data. However, these are not well established and some of them are not even known to the broad machine learning community.

Currently, there are only few papers that provide an overview of methods for handling categorical variables with a comparison of their predictive accuracy. Often, a new method is proposed and compared to approaches that do not account for group membership (Pargent, Pfisterer, Thomas and Bischl, 2021; Johannemann et al., 2019; Yuan & Lin, 2006). Therefore, this thesis assembles a broader set of different existing methods that are designed to recover as much information as possible from categorical variables. Thereupon, the predictive accuracy is compared in several simulations and with real data to show that they achieve a lower estimation error with high-cardinal categorical data than conventional algorithms like OLS or lasso. Moreover, a super learner for data with categorical variables is compiled from the presented algorithms. Overall, this thesis aims to encourage the reader to consider methods that are not wasteful when dealing with categorical variables.

Against this background, the paper continues as follows: In the next section, various methods for handling categorical variables in supervised machine learning are summarized from the literature. Thereby, we give a rationale for choosing the methods that are covered in the thesis. Section 3 explains categorical variables and how to include them in a numerical algorithm. Then, in section 4, three encoding methods and

three regression methods that account for group membership are presented. Section 5 explains how these methods are assembled into a categorical super learner. The predictive accuracy of all methods is then compared to each other and to conventional algorithms in section 6. The thesis concludes with section 7, which provides recommendations for future research.

2 Related work

This section lists existing methods for machine learning that consider the structure of categorical predictors and provides a rationale as to why a method is addressed in the present thesis. These methods cover encoding techniques that provide numerical representations of categorical variables in a lossless manner as well as regression methods that account for groups of variables.

Johannemann et al. (2019) map categorical variables with high-cardinality to a lower-dimensional space without losing any predictive information. Based on their assumption about latent states, encoding methods are created that extract sufficient representations of categorical predictors when used with machine learning algorithms. (pp. 14–15) The proposed low rank encoding method has outperformed other introduced methods in their experiments and is further explained in section 4 (pp. 12–14).

Entity embedding is a yet little studied encoding method that helped Guo & Berkahn (2016) to a podium finish in a kaggle competition. The main idea is to create a mapping of categorical variables into Euclidean spaces, which is learned by a neural network afterwards. That process reveals intrinsic properties of the categorical variable as similar values are mapped close together in the embedding space. (p. 1) Replacing categorical variables with the embedding weights from the neural network improves the prediction accuracy and speed of several machine learning algorithms. (Guo & Berkahn, 2016, p. 1). In the present thesis, entity embedding is used to encode categorical variables in the comparison.

A benchmark experiment on different encoding techniques was conducted by Pargent et al. (2021), which showed that a regularized version of target encoding consistently provides the best results on high-cardinality features compared to traditional strategies. Their target encoder, that combines simple linear mixed models with cross-validation improved the accuracy with any employed regression method and was therefore borrowed for the analysis in this thesis. (p.23)

Heiler and Mareckova (2021) have introduced a flexible penalization approach called pairwise cross-smoothing to handle categorical variables. In this approach, the loss function is penalized by adding sums of weighted squared l_2 -norm differences

between group location parameters and informative first stage estimates. (p. 161) In other words, it uses information from all other columns to penalize the estimate of each column. By the time, this method is not fully developed, as it only works with data that purely consist of categorical predictors. Additionally, pairwise cross-smoothing demands an encoding technique, that creates one dummy variable for every unique observation in the dataset. That is only practical if the dataset has only a few categorical columns with a small number of levels. Otherwise, the data set becomes sparse and needs a lot of observations, since every dummy column should contain at least two occurrences. This context is not applicable for this study since we investigate datasets that contain categorical predictors with many levels.

The least absolute shrinkage and selection operator (lasso) of Tibshirani (1996) has been known for some time and modified in various ways. Some generalizations of the lasso take groups of variables into account, such as one-hot encoded categorical predictors. The group lasso introduced by Yuan & Lin (2006) sets entire groups to zero and thus accounts for properties of categorical variables. This approach has also been extended as it still has some limitations. For example, a standardized group lasso by Simon & Tibshirani (2012) shows improved performance but was not included in the experiments in section 6 because the program crashes when new levels appear in the test set during cross-validation. The sparse-group lasso combines the group lasso penalty with a lasso penalty to also allow the exclusion of predictors from non-zero groups (Simon, Friedman, Hastie & Tibshirani, 2013, p. 239). By penalizing across and within groups, it should outperform the group lasso and lasso in cases where many categorical predictors have many feature levels (Simon et al., 2013, p. 239). An additional extension of the lasso included in the experiment of the present paper is the fused lasso of Tibshirani, Saunders, Rosset, Zhu, & Knight (2005). This algorithm works slightly differently than the other mentioned lasso variants, as it penalizes neighboring predictors in the model rather than groups (p. 91). Since one-hot encoded categorical variables are adjacent in the design matrix, this method may work well on ordinal predictors. Kim (2014) showed that different variants of the lasso with different penalty conditions have advantages depending on the sparsity pattern of the regression parameters (pp. 356–359). Simon et al. (2013) also find that either the lasso, group lasso, or sparse-group lasso work best in their preferred setting with categorical data (p. 239). Although there are many generalizations of the lasso, we chose to include the group lasso, sparse-group lasso, and fused lasso into the

study because together they cover different structures of categorical data with respect to the number of variables and feature levels.

Worth mentioning, but not represented in the thesis, is an implementation of the gradient boosting decision tree algorithm called lightGBM. This implementation uses a technique called exclusive feature bundling that provides a nearly lossless approach to reduce the number of features what makes it 20 times faster than XGBoost (Ke et al., 2017, pp. 1–2). Another advantage is the fact that it handles categorical features according to Fisher (1958) by sorting the feature levels according to the training objective and finding the optimal split over the categories on the sorted histogram (LightGBM, n.d.). Since the method is very complex and has a lot of hyperparameters whose optimization needs much computation, the method is not included in the analysis. Another reason for not considering the tree-based method is that we focus on linear methods because of the scope of the paper and comparability issues.

3 Theoretical Background

Before explaining the methods selected in the last section in detail, the basics of supervised machine learning are recalled. In addition, the properties and challenges of categorical variables are discussed.

3.1 Supervised Machine Learning

The practical part of this thesis in section 6 deals with regression problems, where different algorithms are applied to data and their prediction accuracy is evaluated. The structure of data for this kind of task is having a variable of interest Y also referred to as output, target or dependent variable that is influenced by a set of variables X also called input, predictors, or explanatory variables. Observing data on Y and X , the general problem of supervised machine learning is to approximate the mapping

$$Y = f(X), \quad (1)$$

that represents the impact of each predictor X on the outcome. Once an approximation $Y = \hat{f}(X)$ has been estimated, predictions of \hat{Y} can be made on new data with the same structure as X but an unknown outcome. A practical example would be to learn a mapping or regression model between the sale price of used cars and attributes like the model, age, or mileage to estimate the sale price of other used cars. A simple and concise way to solve such a problem is the ordinary least squares (OLS). It relates the output Y to the input X via the linear model

$$Y_i = \beta_0 + \sum_{j=1}^p X_{ij}\beta_j + u_i. \quad (2)$$

The variable β_0 is the intercept. By including the intercept into the vector of coefficients β and adding a vector of ones to the matrix X , (2) can be written as

$$y = X\beta + u, \quad (3)$$

which is the vector form, and u is the error term representing any factor that affects Y while not being included in X (Hastie, Tibshirani & Friedman, 2009, p. 11). Given the assumption $\mathbb{E}[u] = 0$, the estimated coefficients $\hat{\beta}$ can be computed by minimizing the sum of squared residuals $\sum_{i=1}^N u_i^2 = \sum_i^N (Y_i - \beta_0 - \sum_{j=1}^p X_{ij}\beta_j)^2$, that is equivalent to our notation

$$\hat{\beta}^{ols} = \min_{\beta} \left\{ \|y - X\beta\|_2^2 \right\}, \quad (4)$$

where $\|y - X\beta\|_2^2$ is a squared l_2 -norm that denotes the sum of squares (Hastie et al., 2009, p. 12). This notation will be used throughout the thesis. The regression coefficients can then be used in combination with new input variables X^{new} to make the predictions $\hat{y} = X^{new}\hat{\beta}$. Having reviewed the principles of supervised machine learning and their oldest but still efficient application, the next section sets the focus on input variables of our interest.

3.2 Categorical Variables

Looking at the equations of the previous section, it is obvious that they only work with numerical or quantitative variables. In practice, however, many variables are not continuous. What such discrete variables constitute and how to include them in a model will be explained in the following.

3.2.1 Characteristics of Categorical Variables

A qualitative predictor can be considered as a variable with a set of categories as possible values (Tutz, 2011, p. 1). We will call these sets categorical variables and the possible values feature levels. Categorical variables come in different scaling levels. Nominal categorical variables like colors or zip-codes have no particular order. Even if categorical feature levels come in the form of numbers, they should be understood only as interchangeable labels and statistical analysis should not depend on their order. (Tutz, 2011, p. 4). Often the levels of categorical variables are ordered by nature. Examples for such categories are small, medium, and large. Although these have an order, a metric concept is not applicable, because the distance from small to medium has not to be the same like from medium to large (Hastie et al., 2009, p. 9). The order of so-called ordinal variables may be useful in a data cleaning procedure, by fusing

adjacent categories, when some categories have too few instances. Such a procedure would be very sensible with nominal categorical variables (Tutz, 2011, p. 9).

3.2.2 Encoding of Categorical Predictors

Since most machine learning algorithms require numerical input variables, categorical variables must be transformed accordingly before performing an algorithm. The most simple encoding strategy is label encoding. This is done by replacing each group with a corresponding number and maintaining this numbering throughout the feature level. An example is illustrated in Figure 1. Although this method makes the algorithms run, it entails many problems, even with ordinal categorical variables. A linear predictor would falsely assume a fixed ordering with distances between feature levels being equally spaced (Tutz, 2011, p. 9). After all, C is not equal to 3 times A, as would be implied by the label encoded data in the following example:

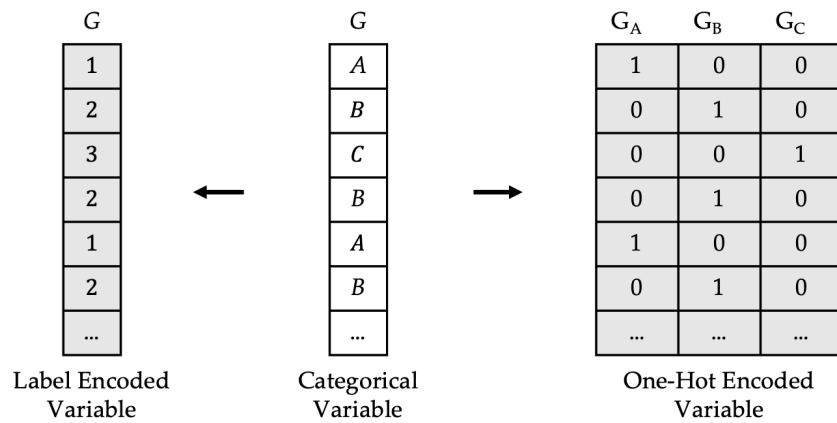


Figure 1: Example of label encoding and one-hot encoding

The better and most commonly used strategy to encode categorical variables is one-hot encoding. It involves creating an additional binary variable for each feature level. The new variable contains a 1 if the observation belongs to this group and 0 otherwise. When the regression model has an intercept, one of the dummy variables is omitted and considered as reference category for the others (Tutz, 2011, p. 16). An advantage of one-hot encoding is that algorithms will consider each feature level to be equally important, which is particularly eminent for nominal categorical variables. However, the encoding strategy still bears some downsides.

3.2.3 Pitfalls of One-Hot Encoding

When applying one-hot coding to categorical data, the dimensionality of the data may grow very high. This can be challenging for algorithms, especially when dealing

with high-cardinality categorical variables such as postal codes that have many rare occurrences (Pargent et al, 2021, p. 2). After encoding, the data set becomes sparse, which means that each column contains mostly zeros (Johannemann et al., 2019, p. 4). As a result, multicollinearity may occur. Apart from that, at least a few samples are required for each feature combination to achieve good results with machine learning algorithms. Furthermore, high dimensional problems require a lot of memory, which slows down computations (Guo & Berkhahn, 2016, p. 1).

The second problem that arises is very much related to the advantage of one-hot encoding. While the independence between feature levels in terms of scaling is desirable, it comes with the loss of informative relation between them (Guo & Berkhahn, 2016, p. 1). Thus, an algorithm assumes as little context between the colors red and green as it does between the color red and the size small. This leads to a loss of information that could be useful for model estimation.

Finding solutions to the loss of group information is the subject of this paper and will be discussed in more detail in Section 4. However, there is an established solution to the high dimensionality problem.

3.2.4 Solution to the Dimensionality Problem of One-Hot Encoding

A commonly applied approach to circumvent at least one of the two problems resulting from one-hot encoding is shrinkage by penalization (Huang & Montoya, 2020, p. 6). The least absolute shrinkage and selection operator, briefly lasso, was proposed by Tibshirani (1996), and became a popular tool to reduce the dimensionality of data by shrinking some coefficients of the model and setting others to zero (p. 267). This has two advantages over OLS: First, by excluding less important columns, the model becomes more parsimonious and interpretable. Second, shrinking reduces the variance at the expense of little bias, which improves the overall prediction (p. 267).

To realize the shrinkage, an L_1 -Penalty is added to the optimization problem (4) of the OLS, that corresponds to the sum of the absolute values of the coefficients. The estimates for the lasso are then computed as follows:

$$\hat{\beta}^{\text{lasso}} = \min_{\beta} \left\{ \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}, \quad (5)$$

where $\lambda \geq 0$ is the penalty parameter and $\|\beta\|_1 = \sum_{j=1}^p |\beta_j|$ is the l_1 -norm.

If $\lambda = 0$, the problem turns into the least squares. As λ grows larger, more penalty is applied, causing the parameters to shrink more towards zero and thus smaller models are chosen. A too large penalty parameter can therefore lead to underfitting. The value of λ is usually determined by cross-validation (CV), where prediction errors for a sequence of possible values is computed. Typically, the smallest model is used that has a prediction error within one standard deviation from the lowest CV-error (Bilder & Loughin, 2015, p. 278).

Even though the lasso has many advantages, there are also limitations. As a variable selection method, it is restricted to n variables. If there are more columns than observations ($p > n$), lasso selects at most p variables as non-zero, even if all variables are relevant (Tutz, 2011, p. 154). With many one-hot encoded high-cardinality variables, such a case may occur eventually.

Like stated above, the lasso can solve the problem of high dimensionality after applying one-hot-encoding to a categorical variable. Nevertheless, there is still the problem of missing coherence between dummy columns describing the same predictor. Moreover, lasso excludes individual dummy variables instead of the whole categorical predictor, although a sensible procedure should consider whole factors (Tutz, 2011, p. 153).

Since lasso is a more stable alternative to OLS for one-hot encoded categorical variables with many factor levels and yet a relatively simple algorithm, it is used in the experiment in section 6 as a benchmark for the encoding methods and algorithms presented next.

4 Dealing with Categorical Variables

Since one-hot encoding of categorical variables followed by lasso only addresses the problem of high dimensionality, this section will first explain three encoding methods and subsequently three generalizations of the lasso algorithm that account for the group membership of one-hot encoded columns.

4.1 Encoding Procedures preserving Group Membership

In the following paragraphs different ways to encode categorical variables are elaborated. In contrast to one-hot encoding, the methods are designed to preserve the group membership of feature levels and are therefore not wasteful. Basically, a nominal categorical variable is substituted by one or more numerical columns that contain all the information from the original column.

4.1.1 Low Rank Encoding

Low rank encoding is an encoding method, that makes use of the other explanatory variables in the model to capture the information of the targeted categorical variable. The method was introduced by Johannemann et al. (2019) in the paper named sufficient representation of categorical variables and is based on an assumption about a sufficient latent state (p. 14). Before facing the encoding method in detail, the underlying premise will be examined.

To represent the group membership, the authors seek a mapping ψ that embeds the group membership in a k -dimensional space with no loss of its predictive information (Johannemann et al., 2019, p. 2):

$$\psi : \mathcal{G} \rightarrow \mathbb{R}^k, \quad \mu(x, g) = f(x, \psi(g)) \quad (6)$$

To realize such a mapping, an assumption about the relationship between the categorical variable and the output is needed. That key assumption is depicted in figure 2: outcome Y_i depends only on the observed categorical variable G_i through an

unobservable latent variable L_i . Hence, if the value of L_i was known, G_i would not provide any further information about Y_i . (Johannemann et al., 2019, pp. 2–4)

For a better understanding, Johannemann et. al (2019) come up with a suitable example about the health status and the choice of a hospital: “A patients health status ($L_i \in \{good, poor\}$) may simultaneously determine to which hospital they are admitted (G_i), what symptoms (X_i) they exhibit and what health outcomes (Y_i) they attain.” (p. 4) In other words, the patient may be admitted to a specific hospital based on the severity of the disease, that influences the health outcome.

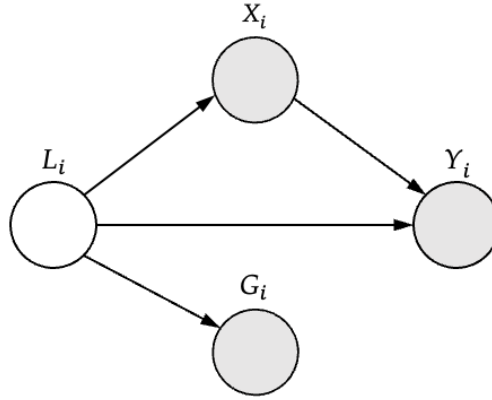


Figure 2: Illustration of the sufficient latent state assumption: X_i and Y_i are only connected to group membership G_i through the latent state L_i which is not observed (own figure based on Johannemann et al., 2019, p. 2)

To derive a sufficient representation of a categorical variable, it is important to understand how the information regarding the categorical predictor G_i comes into the model. Johannemann et al. (2019) explain the context in the following lemma (p. 5):

Suppose that a latent state L_i is discrete with k possible levels, and that the probabilistic structure required by the sufficient latent state assumption holds. Then,

$$\psi: \mathcal{G} \rightarrow \mathbb{R}^k, \quad \psi_l(g) = \mathbb{P}[L_i = l \mid G_i = g] \quad (7)$$

provides a sufficient representation of G_i in the sense of (6):

$$\mu(x, g) = \frac{\sum_{l=1}^k \mathbb{E}[Y_i \mid X_i = x, L_i = l] \mathbb{P}[X_i = x \mid L_i = l] \psi_l(g)}{\sum_{l=1}^k \mathbb{P}[X_i = x \mid L_i = l] \psi_l(g)} \quad (8)$$

Given that k latent groups are present, each feature level can be expressed in terms of one k -dimensional vector of probabilities ψ through which the information about the categorical variable is included into the conditional probability. However, the

estimation of $\psi_l(g) = \mathbb{P}[L_i = l \mid G_i = g]$ is not possible since it depends on the unobservable latent variable L_i . As an alternative, the authors seek functions that depend on observables but can be written as functions of $\psi(g)$ to prove that they are also sufficient representations. (Johannemann et al., 2019, p. 5)

A function that exploits that structure is $f(g) = \mathbb{E}[X_i \mid G_i = g]$, which results in the group-wise means of the numerical variables and can be proven to encode the full information of the categorical variable according to (8) (Johannemann et al., 2019, p. 5). This is also referred to as means encoding when substituting the levels of the categorical variable by these means. The intuition behind it is that the group-wise means are able to uncover the dominant latent group in each level of the categorical variable. (Johannemann et al., 2019, p. 6) This connection is illustrated in figure 3.

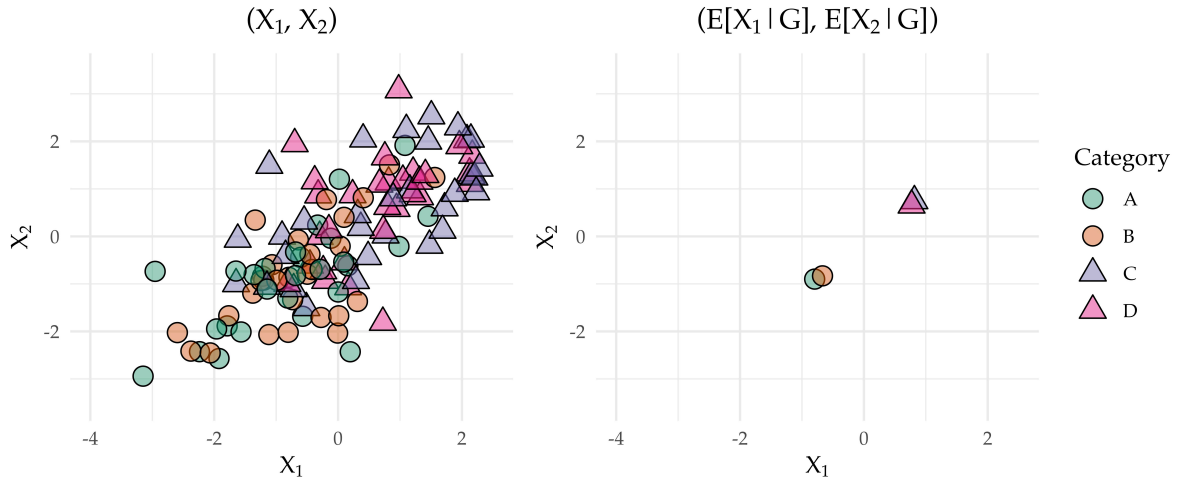


Figure 3: Intuition behind the group-wise means for exemplary data. Separate latent groups are associated with categories (A, B) and (C, D). (own figure based on Johannemann et al., 2019, p. 6)

A drawback of the means encoding method is that a categorical variable may be encoded with more new columns than the number of original levels when there are more continuous predictors present in the data than the cardinality of the categorical variable. This would result in a higher-dimensional encoded variable compared to traditional one-hot encoding. Hence, instead of assigning the groupwise means directly to the levels of the categorical variables, an additional step is implemented, that gives the method the name low rank encoding. In that step, a lower-dimensional representation of the conditional means is created using matrix factorization. (Johannemann et al., 2019, p. 7)

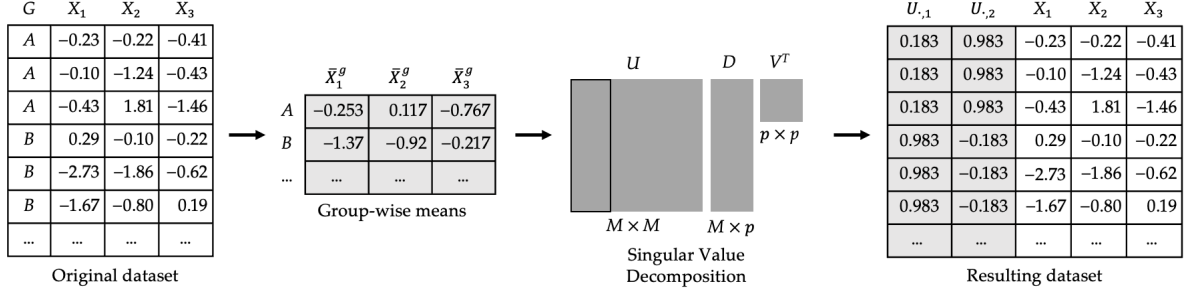


Figure 4: Application scheme for low rank encoding using singular value decomposition (own figure based on Johannemann et al., 2019, p. 7)

Johannemann et al. (2019) factorize the transposed group-wise means matrix Ω using singular value decomposition $\Omega^T = UDV^T$ (p. 7). The first k columns of the g^{th} row of matrix U then represent the g^{th} feature level of the categorical variable (p. 7). The number of required columns k corresponds to the number of latent groups in the data, which would be two in the example in figure 3 (p. 8). In practice k is generally unknown and has to be determined using cross-validation (p. 7). The complete algorithm for the low-rank coding method is provided in Table 1.

Algorithm: Low Rank Encoding

Require $k \in \mathbb{N}$

1. Calculate group-wise means $\hat{\Omega}$ of continuous covariates X_i for each level g

$$\hat{\Omega} \leftarrow 0_{p \times M}$$

for g in 1:M **do**

$$\hat{\Omega}_{:,g} \leftarrow \frac{1}{|\{i: G_i = g\}|} \sum_{i: G_i = g} X_i$$

2. Singular value decomposition of group averages

$$U, D, V^T \leftarrow SVD(\hat{\Omega}^T)$$

3. Populate $S \leftarrow 0_{n \times k}$ with left singular matrix truncated rows

for i in 1:n **do**

$$S_{i,\cdot} \leftarrow U_{G_i, 1:k}$$

Notation: G is a single categorical variable and g is a feature level of G ; the nominal variable G is encoded into the first k numerical columns S of the left singular vector matrix U ; value k is found by cross-validation.

Table 1: Algorithm of low rank encoding (Johannemann et al., 2019, pp. 7–8)

In the paper sufficient representation of categorical variables, the authors apply low rank encoding on the categorical variable with the most feature levels both in

simulation and with real data. That categorical predictor is respectively encoded, and the other nominal categorical variables enter the model as numerical variables, using label encoding (see section 3.2.2). This is necessary to the extent that the calculations are based on numerical values in every other column. Nevertheless, the procedure is questionable because it is calculated with numbers that are to be understood as mere labels. Taking that into account, the best setup for low rank encoding would be a dataset that contains only one categorical variable beside numerical variables. Consequently, it would not be a problem to evaluate the encoded data with the OLS, since no label encoded data would be included in the resulting dataset.

It is unfortunate that no more than one categorical variable can be low-rank encoded. Perhaps there would be a way to code more than one categorical variable by calculating the low-rank representation for each categorical variable separately, and then replacing the categorical variables in the original data set with the corresponding representations once all categorical variables have been encoded. This could be interesting for future investigations.

4.1.2 Entity Embedding

Similar to low rank encoding, entity embedding encodes categorical variables in a meaningful way in a lower dimensional space. In fact, embeddings for words were formerly learned using singular value decomposition (Deerwester, Dumais, Furnas, Landauer & Harshmann, 1990, pp. 395–399). This idea was further pursued, and the word embeddings would later be learned using neural nets where similar words are placed closer together in a word space (Bengio, Ducharme, Vincent & Jauvin, 2003, p. 1137). As a consequence, neural nets became popular in speech recognition, computer vision and natural language processing, which are all problems given unstructured data¹ (Guo & Berkhahn, 2016, p. 1). As neural networks only process numerical data, they have limited applicability to categorical variables that are often found in structured data, such as those studied in the present thesis (Guo & Berkhahn, 2016, p. 1). Therefore, naive application of neural networks to structured data with label encoded categorical variables does not work well, and with one-hot encoding the problems described in Section 3.2.3 arise (Guo & Berkhahn, 2016, p. 1). In the case of

¹ Unstructured data is not formatted in a database structure with columns and observations. Imaging, audio, and text data are examples of unstructured data.

neural networks, the loss of group membership of one-hot encoded variables can be solved using entity embedding of categorical variables which is similar to the word embedding method mentioned above. Entity embedding learns the representation of feature levels in multidimensional spaces, where values with similar effects are placed close together, and thus exposes the underlying continuity of the data and helps the neural network solving that problem (Guo & Berkhahn, 2016, p. 1). Moreover, the resulting vector representations of the feature levels can be used as replacement of the categorical predictors for other common machine learning algorithms instead of one-hot encoding. The next section explains in more detail how embeddings are produced.

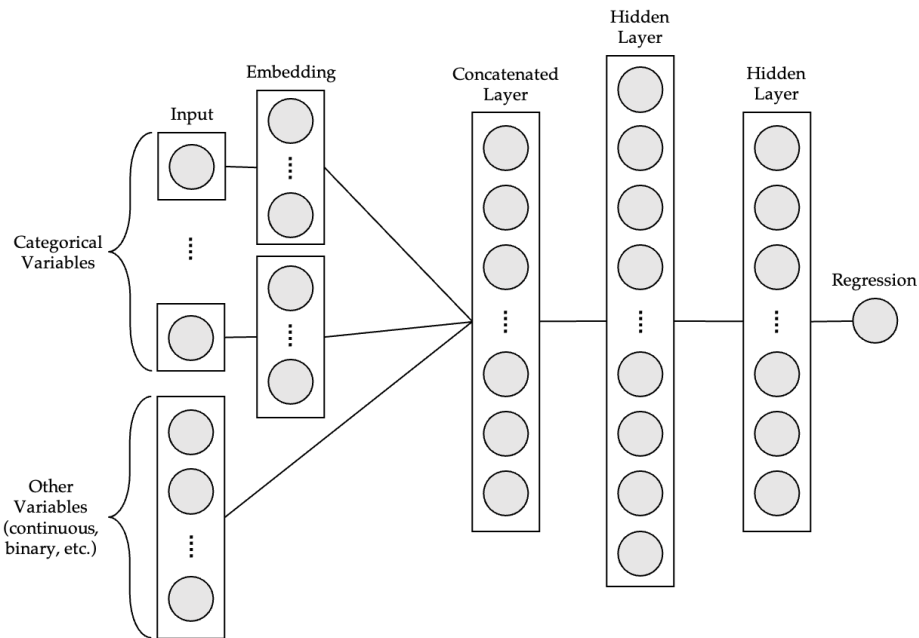


Figure 5: A basic neural network architecture with embeddings (own figure based on Forlenza, Lapiello & Emilio (2019))

Entity embedding maps the feature levels of a categorical predictor onto a vector of dimension $1 \times d$. The mapping corresponds to an additional layer of d neurons built on top of the input layer, which contains the one-hot encoded categorical variables. That layer serves as a lookup table for its weight matrix E with $C \times d$ dimensions. Note that C represents the number of feature levels, and d is a hyperparameter, that is chosen to obtain a compacter representation of the categorical variable, while maintaining any relevant information. The row-vectors of the embedding matrix are concatenated with the other continuous inputs and are then fed to the rest of the network, where additional hidden layers can be built on top of it according to figure 5. Along

with the other parameters, the weights are initialized randomly and optimized during the back propagation of errors². (Mezzogori & Zammori, 2019, p. 1797)

As a result, the underlying properties of the feature levels are learned by the embedding layer, whereas complex combinations of them are formed at deeper layers. After training the neural network, the learned weights of the embedding layer can be extracted and used to encode categorical variables. In order to achieve that, the embedding vector of dimension $1 \times d$ for each category feature is used to replace the corresponding level of the categorical variable in the data. As a result, d numerical columns are created, which describe the original categorical column sufficiently (see figure 6). (Guo & Berkahn, 2016, p. 4)

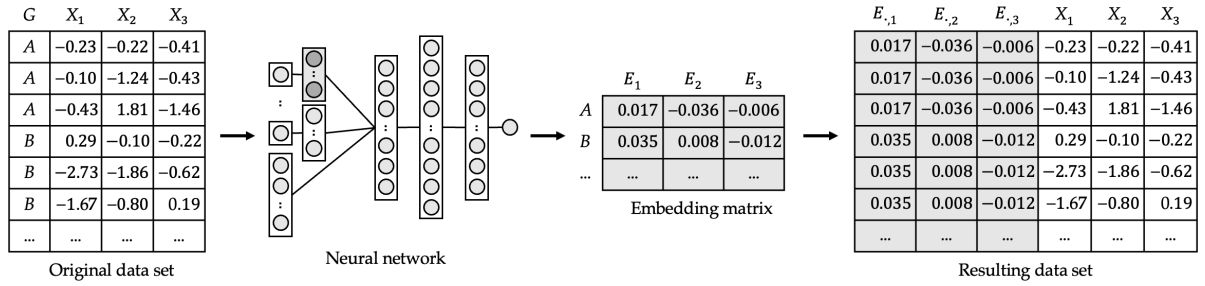


Figure 6: Implementation example of entity embedding with three output columns

A formal description of neural networks is not provided in this thesis to avoid straying too far from the actual topic. However, we will point out some challenges and drawbacks of neural networks and entity embedding. According to Hastie et al. (2009), training a neural network is quite an art since we have a nonconvex and unstable optimization problem, and a model that is generally overparametrized (p. 397). A poorly trained neural network can lead to overfitting, stagnation of the learning process, or widely varying weights. Therefore, the user is advised to find the optimal number of hidden layers as well as the number of neurons, the preferred activation function and regularization for each layer. Furthermore, the updating algorithm with a suitable learning rate as well as the number of epochs and the mini-batch size should be specified. As a consequence, hyperparameter tuning of a neural network is a time-consuming procedure even when certain guidelines are followed.

² Back-Propagation is a learning procedure used in neural networks that minimizes a difference measure (e.g., mean-squared error) between the desired output and the output vector of the net by repeatedly adjusting the connection weights within the network. The resulting weights in combination with hidden layers represent important features in the data. (Rumelhart, Hinton & Williams, 1986, p. 533)

Nevertheless, it is worth the effort, since neural networks tend to outperform other machine learning techniques (Guo & Berkahn, 2016, p. 7). Furthermore, Guo & Berkahn (2016) show that entity embeddings learned from a neural network improve the performance of other machine learning techniques when used as an input rather than one-hot encoded categorical variables (pp. 6–7).

4.1.3 Regularized Target Encoding

Target Encoding is a method that uses information about the dependent variable associated with a given level of a categorical predictor (Pargent et al., 2021, p. 3). Therefore, it is a supervised encoding technique, which is well suited for cases where the predictor has many possible levels or when new levels appear after training (Kuhn & Johnson, 2019, para. 5.4). The main idea is to use the training set to make a prediction of the target for each level of the categorical variable and use the prediction as a numerical value for the respective level (Pargent et al., 2021, p. 3). Hence, it is related to the method weight of evidence of Hand & Henley (1997), which has been used in credit scoring (p. 527).

In the simplest form of target encoding, the mean of the dependent variable of all observations with the same feature level is used in place of that level. However, this method is not optimal because the conditional expectation of the target given a level is similar to the true target, which leads to overfitting. This effect can be weakened by excluding the target of the current row in the calculation of the mean target, which is called leave-one-out encoding. A better way to reduce overfitting is to use regularized target encoding by adding a smoothing parameter, which shrinks the effects toward the global mean (Micci-Bereca, 2001, p. 28). An alternative strategy is to use cross-validation to combine target encoding of each fold (Pargent et al., 2021, p. 4).

The regularized target encoding method used in the mentioned paper takes advantage of both of these measures. According to Kuhn and Johnson (2019), shrinkage can also be accomplished by general linear mixed models in their estimation procedures (para. 5.4). A mixed model is a statistical model that contains both fixed effects and random effects (Pargent et al., 2021, p. 9):

$$Y_i = \underbrace{X_i\beta}_{\text{fixed}} + \underbrace{Z_i\mu_i + \epsilon_i}_{\text{random}} \quad (9)$$

Fixed effects are unknown constant parameters associated with continuous covariates or the levels of the categorical variables and can be seen as regression coefficients.

Random effects on the other hand, are represented by unobserved random variables which are usually assumed to follow a normal distribution. (West, Welch & Galezki, 2014, p. 1)

In the given approach, not a full mixed model is applied on the training set, but a random intercept model, that only makes use of the fixed intercept and the random intercept, which is allowed to vary for each level l of the categorical predictor (Pargent et al., 2021, p. 9):

$$Y_i = \beta_{0l} + \epsilon_i = \underbrace{\gamma_{00}}_{\text{fixed intercept}} + \underbrace{u_l}_{\text{random intercept}} + \epsilon_i \quad (10)$$

Pargent et al. (2021) carry out conditional estimates for each level of a categorical predictor in a random intercept model and use it as encoded value for that level. That is similar to the mean target value for each variable, weighted by the relative observed frequency of that level in the training set. In this procedure, the target is predicted by the sum β_{0l} of a random intercept u_l for each level of the categorical variable plus a fixed intercept γ_{00} . The fixed intercept can additionally be used to encode levels in the test set, that are not observed within the training set. Another advantage of the linear mixed model is that it does not require a tuning parameter, because a reasonable amount of regularization is determined automatically. (p. 8) To further avoid overfitting, Pargent et al. (2021) combine target encoding on linear mixed models with cross-validation. Specifically, they fit a linear mixed model on each resulting training set and then use the estimated intercept for the observations that are not used for model fitting as encoded values. In the test set, feature values are always encoded by a single linear mixed model fitted on the complete training set. (p. 8) The complete algorithm can be followed on table 2.

Pargent et al. (2021) show in their conducted benchmark study on 24 publicly available datasets with high-cardinality features, that regularized target encoding tends to outperform other encoding methods³ in combination with each tested machine learning algorithm⁴ on regression and classification problems (pp. 14–20).

³ The other tested encoding methods were integer, frequency, one-hot, hash, leaf, and impact encoding.

⁴ Lasso, regression forest, gradient boosting, support vector machines and k-nearest neighbors are the machine learning algorithms used in the paper (Pargent et al., 2021, pp. 14–20).

Algorithm: GLMM Encoding Regression

Training set: require $n.folds \in \mathbb{N}$

1. fit a simple intercept model: $y_i^{train} = \beta_{0l} + \epsilon_i = \gamma_{00} + u_l + \epsilon_i$ on \mathcal{D}^{train}
with $u_l \sim N(0, \tau^2)$, $\epsilon_i \sim N(0, \sigma^2)$ and $x_i^{train} = l$, $l \in \mathcal{L}^{train}$
2. use $n.folds$ cross-validation scheme to make training sets $\mathcal{D}_1^{train}, \dots, \mathcal{D}_{n.folds}^{train}$
and fit simple random intercept model on each \mathcal{D}_m^{train}
for all $x_i^{train} \in \mathcal{L}^{train}$ **do**
 $\hat{x}_i^{train} = \hat{\beta}_{0l}^{\mathcal{D}_m^{train}}$ with $x_i^{train} = l$ based on the model m with $(x_i^{train}, y_i^{train}) \notin \mathcal{D}_m^{train}$

Test set:

- for** x^{test} **do**
if $x^{test} \in \mathcal{L}^{train}$ **then**
 $\hat{x}^{new} = \hat{\beta}_{0l}^{\mathcal{D}_m^{train}}$ with $x^{test} = l$ based on full model fitted on \mathcal{D}^{train}
else $\hat{x}^{test} = \hat{\gamma}_{00}$ based on full model fitted on \mathcal{D}^{train}
-

Notation: x is a single level of a categorical variable; the nominal feature x^{train} is transformed into the numerical feature \hat{x}^{train} , which is used for training. We use \hat{x}_i as the encoded value for an observation with level $l \in \mathcal{L}$.

Table 2: Algorithm of GLMM encoding regression (Pargent et al., 2021, p. 9)

4.2 Regression methods with Group Penalization

In contrast to the last section, which only covered the encoding of categorical variables, this section discusses complete machine learning algorithms. Although these cannot be fed directly with categorical variables, they jointly shrink multiple columns created by an encoding method such as one-hot encoding. In this way, the group membership can be preserved in the regression. All algorithms presented below are generalizations of the lasso (see section 3.3.4).

4.2.1 Group Lasso

As an extension of the lasso, Yuan & Lin (2006) introduced the group lasso that selects entire groups of variables instead of individual variables (p. 51). These groups can be predefined exogenously, such as the one-hot encoded levels of categorical variables. Continuous or binary variables have group sizes of one. A group of variables may also refer to interactions of factors with continuous variables or between factors, where the group size is the number of interactions (Tutz, 2011, pp. 153–154).

Formally, the predictor variables are divided into m different groups, which may be selected by the following minimization problem (Simon et al., 2013, pp. 2150–2151):

$$\hat{\beta}^{\text{GL}} = \min_{\beta} \left\{ \frac{1}{2} \left\| y - \sum_{l=1}^m X^{(l)} \beta^{(l)} \right\|_2^2 + \lambda \sum_{l=1}^m \sqrt{p_l} \|\beta^{(l)}\|_2 \right\}, \quad (11)$$

where $X^{(l)}$ is the submatrix of X whose columns correspond to the one-hot encoded categorical variable denoting group l , $\beta^{(l)}$ is the vector of coefficients of that group and p_l is the length of $\beta^{(l)}$. The minimization problem utilizes the non-differentiability of $\|\beta^{(l)}\|_2$ at $\beta^{(l)} = 0$ by setting entire groups of coefficients to zero. The magnitude of penalty parameter $\lambda \geq 0$ determines the sparsity of the solution. When the size of each group is 1, the solution of the regular lasso is obtained. (Simon et al., 2013, p. 2151)

Yuan & Lin (2006) assume the submatrices $X^{(l)}$ to be orthonormalized⁵ (p. 50). Nevertheless, the statistical community has often adopted the approach for correlated features for which the penalty is less effective (Simon & Tibshirani, 2012, p. 983). Therefore, Simon & Tibshirani (2012) developed a standardized group lasso with a penalty matrix that compares favorably to the unstandardized group lasso (pp. 985, 1001).

Another variant of the group lasso is the modified group lasso, introduced by Choi, Park & Seo (2012). Since the penalty gets bigger as $\sqrt{p_l}$ increases, the group lasso tends to exclude groups with high dimensions (p. 2). The modified group lasso corrects this by removing the parameter $\sqrt{p_l}$ to give all variables the same chance of being selected when other conditions are the same (p. 2). Remarkably, the estimates are more robust to the choice of λ than lasso and group lasso (p. 3). Overall, the method should perform better than the group lasso, particularly for large column dimension with many feature levels (p. 1). However, this could not be confirmed, in our experiments, so the normal group lasso was chosen for the comparison.

⁵ $X^{(l)}$ is orthonormal when $X^{(l)'} X^{(l)} = I_{p_l}$ for $l = 1, \dots, m$. This can be done through Gram-Schmidt Orthonormalization (Yuan & Lin, 2006, p. 50)

The remaining drawback of the group lasso is that when it selects a group of variables, then all variables in that group will be nonzero (Simon et al., 2013, p. 231). But for some problems, we seek both sparsity of the groups and sparsity within groups, in the case when some levels of a categorical variable are particularly important (Simon et al., 2013, p. 231). The regularized regression method that addresses this problem is called sparse-group lasso and is described in the next section.

4.2.2 Sparse-Group Lasso

The sparse-group lasso was created by Simon et al. (2013) as an extension of the group lasso, which provides sparsity of groups as well as sparsity within each group (p. 231). This means that still entire groups are excluded from the model but from the selected groups only individual variables may be selected. For this purpose, a combination of the group lasso penalty and the lasso penalty is applied (Simon et al., 2013, p. 232):

$$\hat{\beta}^{\text{SGL}} = \min_{\beta} \left\{ \frac{1}{2n} \left\| y - \sum_{l=1}^m X^{(l)} \beta^{(l)} \right\|_2^2 + (1 - \alpha) \lambda \sum_{l=1}^m \sqrt{p_l} \|\beta^{(l)}\|_2 + \alpha \lambda \|\beta\|_1 \right\} \quad (12)$$

Here, $\alpha \in [0, 1]$ sets the weights for the lasso and the group lasso. All remaining parameters correspond to those in (5) and (11). Setting $\alpha = 0$ results in the group lasso and setting $\alpha = 1$ results in the lasso. In this property, the method has similarities with the elastic net, which mixes the lasso and the ridge penalty (Zou & Hastie, 2005).

According to Simon et al. (2013), each of the three lasso variants has a preferred data structure. When there are only a few predictors with high-cardinality, the lasso is most suitable, since much information would be lost by excluding whole groups. In the case of many such categorical variables, the sparse-group lasso should be used, since there may be uninformative groups, but also uninformative categories in groups that should be selected. Finally, the group lasso is recommended when there are many categorical variables with only few levels, since several groups may not be informative. (p. 239).

Both group lasso and sparse-group lasso select variables by taking predefined groups into account. The fused lasso, on the other hand, does not consider fixed groups of variables, but still comes in handy when dealing with categorical predictors. More details are given in the next section.

4.2.3 Fused Lasso

The final generalization of the lasso discussed in the present thesis is the fused lasso proposed by Tibshirani, Saunders, Rosset, Zhu, & Knight (2005). With this approach, the order of the parameters in the model is exploited. This requires that the features have a natural order or can be ordered in a meaningful (p. 91). To utilize this order, the fused lasso adds an additional penalty to the normal lasso (5):

$$\hat{\beta}^{\text{FL}} = \min_{\beta} \left\{ \|y - X\beta\|_2^2 + \lambda_1 \sum_{j=1}^p |\beta_j| + \lambda_2 \sum_{j=2}^p |\beta_j - \beta_{j-1}| \right\}. \quad (13)$$

The penalty λ_1 induces sparsity in the coefficients whereas the penalty λ_2 leads to sparsity in their differences, resulting in flattening of coefficient profiles β_j (Tibshirani et al., 2005, p. 93). This means, that adjacent levels may be fused by aligning the coefficients. When the features do not have a natural order, then an order for the features can be estimated using multidimensional scaling or hierarchical clustering (Tibshirani et al., 2005, p. 104). The latter reveals the correlation of the data and assembles the features that have a similar correlation. According to Tibshirani et al. (2005), the order does not have to be complete; it is sufficient when the nearest neighbors are specified (p. 104).

The fused lasso has similar sparsity properties like the lasso (Tibshirani et al., 2005, p. 20). However, while the lasso can select a maximum of N columns, the fused lasso selects a maximum of N sequences with equal non-zero coefficients (Tibshirani et al., 2005, pp. 20–21). This offers an advantage, especially for problems with $p \gg N$, for which the fused lasso was originally designed. In that type of setting, the fused lasso performed better than the normal lasso in the experiment of Tibshirani et al. (2005, p. 24).

Motivated by the fused lasso, Gertheiss and Tutz (2010) proposed a modified penalty, that allows shrinkage and clustering of unordered categorical variables in a regression context (p. 2155). The penalty $\sum_{i>j} w_{ij}^{(l)} |\beta_{li} - \beta_{lj}|$ considers all differences of the dummies in each categorical variable (p. 2155). By setting some differences to zero, the data becomes clustered (p. 2155). That approach delivers nicely interpretable results and reduces the dimensionality of data.

5 Super Learner

One objective of the thesis is to integrate at least two of the methods explained above for handling categorical variables into a super learner. Therefore, this section explains what a super learner is and how it can be implemented. A super learner consisting of the six applications presented will be compared to the other methods in the experiment in section 6.

5.1 Benefits of a Super Learner

When facing a regression problem, a single model is often chosen to solve the problem. But since the most suitable model is usually not known a priori, an algorithm that selects the best model or even combines different models may be convenient. That thought originated from Wolpert (1992) with the goal to improve prediction accuracy (pp. 241–259). Breiman (1996), who was engaged in aggregating regression trees at the time, picked up on this idea, provided a numerical solution and showed that such an ensemble indeed yields an improvement in performance (pp. 49–64). Finally, according to the proof of van der Laan, Polley & Hubbart (2007), for large sample sizes, the accuracy of an ensemble is always better than the best estimator contained in it (pp. 1317–1319). Given that, the best practice would be to choose a large library of machine learning algorithms to improve the performance by creating the optimally weighted combination of them (Naimi & Balzer, 2018, p. 459). The biggest gains occur when dissimilar sets of predictors are stacked (Breiman, 1996, p. 51). Stacking or stacked generalization is the name of the method to which we refer. An intuition of that method and its technicalities will be given in the next section.

5.2 Stacked Generalization

The goal of stacked generalization is to find a weighted combination of a set of algorithms to form a super learner. These weights are computed using non-negative least squares to regress the true outcome against the predictions of each candidate

algorithm (Naimi & Balzer, 2018, pp. 460–461). In its original form, the formal definition of the non-negative least squares states

$$\hat{\beta}^{\text{nnls}} = \min_{\beta \geq 0} \left\{ \frac{1}{n} \|y - X\beta\|_2^2 \right\}, \quad (14)$$

where β would be a vector of weights and X corresponds to the matrix of the predictions of each candidate algorithm in our case. The weights are then constrained to add up to 1 by dividing them by their sum. In other words, the mean squared error $\frac{1}{n} (Y - \hat{Y})^2$ is minimized under the mentioned constraints, which are elaborated in Breiman (1996, pp. 50–53). If the predictions were constructed with the same learning set used to calculate the mean square error, the weights will overfit the data (Breiman, 1996, p. 50). To overcome that problem, the predictions are formed using V-fold cross-validation⁶ (Naimi & Balzer, 2018, p. 460). The resulting weights are then used for a convex combination of the predictions of the candidate algorithms fitted on the entire sample (Naimi & Balzer, 2018, p. 461). The algorithm is illustrated in detail in table 3.

Algorithm: Super Learner

1. Use cross-validation to create training sets $\mathcal{D}_1^{\text{train}}, \dots, \mathcal{D}_{n, \text{folds}}^{\text{train}}$ and fit each candidate algorithm on each $\mathcal{D}_v^{\text{train}}$ to get the cross-validated predictions $\hat{Y}_1^{cv}, \dots, \hat{Y}_n^{cv}$.
 2. Apply the non-negative least squares of the actual outcome Y on the cross-validated predictions from step 1, constraining the coefficients sum up to 1 to get the weights of the optimal predictor:
 $\mathbb{E}(Y | \hat{Y}_1^{cv}, \dots, \hat{Y}_n^{cv}) = \alpha_1 \hat{Y}_1^{cv} + \dots + \alpha_n \hat{Y}_n^{cv}$, where $\alpha_1, \dots, \alpha_n \geq 0$ and $\sum_{k=1}^n \alpha_k = 1$
 3. Refit the candidate algorithms on the entire sample and combine the predictions with the estimated weights from step 2: $\hat{Y}_{SL} = \alpha_1 \hat{Y}_1 + \dots + \alpha_n \hat{Y}_n$.
-

Table 3: Algorithm of a super learner (Naimi & Balzer, 2018, pp. 460–461)

⁶ The term V-fold cross-validation used in Naimi & Balzer (2018) and in van der Laan et al. (2007) is equivalent to K-fold cross-validation.

An improvement of the Super Learner's predictions may be achieved with a large set of candidate algorithms. A drawback of this, however, is the computation time. Since a V-fold cross-validation is necessary, the computation takes a multiple compared to the single algorithms. This is especially noticeable with algorithms that already require a lot of computing power themselves. Thereby, it is still important to perform hyperparameter tuning, which requires additional cross-validation for some methods (Naimi & Balzer, 2018, p. 462). This may result in very small folds being used for cross-validation in the tuning process, which can lead to inaccuracies. As a solution, an algorithm can be included into an ensemble learner several times with various parameters (Naimi & Balzer, 2018, p. 462). Another limitation of the super learner is that it is unclear how each covariate exactly contributes to the prediction. (Naimi & Balzer, 2018, p. 463).

5.3 Including Algorithms into a Super Learner

As mentioned at the beginning, a part of the thesis involves the implementation of two presented algorithms into an ensemble R-function being developed by the supervisor of this thesis⁷. The algorithms chosen for implementation are the group lasso and fused lasso. Although the two algorithms perform well, the reason is rather the exclusion of the other methods. First, the sparse-group lasso consumes too much time when working with a large sample. Second, entity embedding is built on functions that rely on python, whose environment is nontrivial to get working. Third, with GLMM, as with entity embedding, the encoding procedure of the test set relies on the encoded data from the training set, which is difficult to implement. Last, low rank encoding did not entirely satisfy the author. Not because it performs badly, but because only one categorical predictor is encoded accordingly while the other categorical variables are converted to integers. On the other hand, the fused lasso is easy to implement and so is the group lasso, although with the latter it is more challenging to assign the group memberships if the data is already one-hot encoded or interactions have been added.

⁷ <https://rdrr.io/github/MCKnaus/causalDML/man/ensemble.html>

6 Experiments

The goal of the experiments is to compare the prediction accuracy of the presented approaches from Section 4 among each other and with the ordinary least squares and lasso as benchmarks. First, a simulation with different settings is performed. This involves varying the number of variables and feature levels as well as the true regression coefficients. Then, the methods are applied to real datasets to validate the results of the simulations. Finally, the results are discussed in the last section of this chapter.

6.1 Simulations

A custom approach was coded for the simulation of datasets with categorical variables, in which any number of continuous variables and categorical variables with different feature values can be simulated. Parameters such as the number of observations, the correlation of the covariates, and the signal-to-noise ratio can be determined. Regression coefficients are specified as input variables to compute the response variable. The code used to simulate the data is linked in appendix A, and the procedure is described in the following.

6.1.1 Data Generating Process

Since the data-generating process involves categorical variables, the procedure is more complicated, as the coefficients of the feature levels must also be taken into account to calculate the true outcome. However, the resulting dataset should not contain dummy variables, but categorical variables, since also encoding techniques are performed with the data.

In a first step, the covariates are drawn from a multivariate normal distribution. The categorical variables are firstly treated like continuous variables, but then each data column is binned into equal sized intervals in respect to the desired number of feature levels. The bins are formed with sorted data, so the correlation structure remains valid for the categorical variables. Operatively, this is done by assigning each level l to one of the l quantiles of a continuous variable and then converting the variable into a factor. Johannemann et al. (2019) used a similar approach in their simulation study

(p. 10). In a second step, the coefficients β that also apply to the dummy variables are given as input variables. To finally generate the continuous outcome vector y , a design matrix X^{enc} is built from the data matrix X , where the categorical variables are one-hot encoded. In this way, each coefficient can be assigned to a dummy variable or a numerical variable with the equation

$$y = X^{enc}\beta + \epsilon = \sum_{l=1}^L X_l^{enc}\beta_l + \epsilon. \quad (15)$$

Here, the error term ϵ is adjusted to have a pre-defined signal-to-noise ratio (Choi, Park & Seo, 2012, p. 2; Yuan & Lin., 2006, p. 60). An overview of the data-generating process can be found in table 4:

Data-Generating Process		
Outcome variable (y):	$y = \beta_0 \mathbf{1} + \beta^{(x)}X + \beta^{(c)}C^{OHE} + \epsilon$	(16)
Sampled variables (Z):	$Z \sim N(0, \Sigma)$	(17)
Continuous variables (X):	$X = Z^{(x)}$	(18)
Categorical variables (C):	$C = (\mathcal{Q}_i = i)_{i=1}^L$, where \mathcal{Q}_i is the i^{th} quantile of $Z^{(c)}$	(19)
Noise (ϵ):	$\epsilon \sim N\left(0, \sqrt{\frac{Var(Y)}{SNR}}\right)$	(20)

Notation: L denotes the number of feature levels of a categorical variable. C^{OHE} means the one-hot encoded matrix C . $Z^{(x)}$ is the subset of Z that includes continuous variables and $Z^{(c)}$ is the subset of Z that is transformed to categorical variables. The same logic applies to the betas. The expression in (19) implies setting the values of each quantile of $Z^{(c)}$ to the number of the quantile \mathcal{Q}_i . Finally, SNR stands for signal-to-noise ratio.

Table 4: Data-generating process for the simulations

6.1.2 Simulation Setup

The prediction accuracy of the methods is assessed in two main scenarios that are evaluated with four different settings of the beta coefficients, which will be explained later. Additionally, two more scenarios are implemented that should demonstrate the behavior of the methods under extreme conditions.

In each scenario the sample size is fixed to 300 observations. This size is kept rather small due to time constraints, as many different settings are simulated over 1000 runs. The correlation between each variable is set to 0.3 and the error term is sampled with

a signal-to-noise ratio of 3. Finally, the sampled data is divided to a training and test set in a ratio of 2 to 1 with a stratified split on the categorical variable with the highest cardinality. The entire simulation study procedure is summarized in table 5.

In the main scenarios the data consists of 4 categorical variables and 4 continuous variables. They differ in the number of feature levels of the categorical variables, where scenario 1 has a moderate number of feature levels (8, 5, 4, 3) and in scenario 2, the number of feature levels is rather high (30, 15, 10, 5) compared to the sample size. In addition, two extreme cases are considered: data consisting of only one categorical variable with a cardinality of 80 alongside with some continuous variables, as well as data with 20 categorical variables with only 3 feature levels.

Since the regularized regressions that obtain the group information penalize the beta coefficients, the simulations for each scenario are equipped with beta coefficients that satisfy different properties with the goal to better analyze the behavior of the algorithms. This involves varying the beta coefficients between different categorical variables, that are considered as groups of dummies, and within these groups. Huang & Montoya (2020) did something similar, but in the context of the impact of coding strategies on the lasso and group lasso (pp. 21–24).

- DGP 1: No variance among groups and no variance within groups
- DGP 2: Some variance among groups and no variance within groups
- DGP 3: No variance among groups and some variance within groups
- DGP 4: Some variance among groups and some variance within groups

No variance means zero variance among or within groups of betas. Some variance refers to a variance of 1. The values of the beta coefficients are defined accordingly. Exact values of the coefficients for each simulation can be found in Appendix B.

Simulation Study

1. Set up the data-generating process and define the desired beta coefficients.
 2. Draw a random sample of $N = 300$ observations from the DGP.
 3. Split the sample into a simulation sample of $N_s = 200$ and a validation sample of $N_v = 100$. Then, set the outcome variable Y_v aside.
 4. Train the algorithms on the $N_s = 200$ observations and use X_v from the validation sample to predict \hat{Y}_v .
 5. Calculate the mean-squared error for each prediction and store it in a matrix.
 6. Repeat steps 2 to 5 for $m = 1000$ times.
-

Table 5: Overview of the simulation study

6.2 Real Data


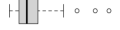


To ensure that the simulation results are consistent in practice, the methods are additionally applied and evaluated on four real datasets that are described in this section.

Avocado Prices is a dataset from Kaggle that shows weekly sales of Hass avocados from 2013 to 2018 broken down by different sales regions and package quantities (Kiggins, 2018). Price per avocado is the target variable.

The *Ames Housing* data was collected by De Cock (2011) and describes home sales in Ames between 2006 and 2010 (p. 1). It includes a large number of categorical predictors and is a more complicated alternative to the popular Boston Housing data (p. 2).

Employee Salaries is a dataset obtained from OpenML (Pargent, 2019). It includes the annual salaries of employees of Montgomery County in 2016. High-cardinality features like the job position makes this dataset interesting for the analysis in this paper.

Cortez & Silva (2008) collected *Student Performance* data from a Portuguese school (p. 2). The data consist of the final grade as well as demographic, social and school related predictors (p. 2). Table 6 provides a descriptive analysis of the datasets:

Name	N	Num	Bin	Cat	HighCardLevels	Entropy
Avocado Prices	18249	8	1	2	54	
Ames Housing	2930	34	2	44	10, 10, 16, 17, 17, 28	
Employee Salaries	9228	1	2	3	37, 385, 694	
Student Performance	649	2	13	15	5, 5, 5, 5, 5, 5, 5, 5, 5	

Note: N = observations, Num = numeric variables, Bin = binary variables, Cat = categorical variables, HighCardLevels denotes the cardinality for each categorical variable with more than 10 levels (5 levels at student performance dataset), Entropy = boxplot of the normalized Shannon-entropy across levels (smaller = larger imbalance).

Table 6: Description of real-world datasets (based on Pargent et al., 2021, p. 11)

The raw data was changed as little as possible. If there was only one or two occurrences of a feature level in ordinal variables like years, it was merged with the closest one. Single feature levels of nominal variables were binned into a new level called "others". This procedure ensures that the test set does not contain feature levels that are not present in the training set. Finally, the data was divided by a stratified split in a ratio of 2 to 1 like in the simulations.

6.3 Implementation of Methods

This section briefly describes how the different methods are implemented in the statistical program R and which tuning-parameters were chosen to improve the performance.

6.3.1 Low Rank Encoding

The implementation of low-rank encoding is based on the code⁸ from Johannemann et al. (2019). However, for simplicity, the wrapper function for low-rank encoding was extracted from the rich function and implemented according to the algorithm in Table 1. Low rank encoding is performed on the categorical variable with the most factor levels and the remaining categorical variables are left as integers. To make the predictions, an OLS regression is conducted on the encoded data.

⁸ <https://github.com/grf-labs/sufrep>

6.3.2 Regularized Target Encoding

GLMM-encoding is implemented with the supplemental code⁹ from Pargent et al. (2021) which follows the steps in table 2. The data is encoded using 5-fold cross-validation and followed by an OLS regression to make the predictions.

6.3.3 Entity Embedding

To implement the neural net, the R package *keras* is used. *Keras* does not run directly in R but on a *tensorflow* backend via Python. Before building the neural net, the continuous variables and the output is scaled. Each categorical variable is one-hot-encoded and placed into an embedding layer with a number of neurons that equals the truncated cube root of the number of levels to reduce the dimensionality. The embedding layers are then concatenated with the continuous inputs. Then, one hidden layer with 500 neurons is added and a final layer with one neuron. To update the neural net, the optimizer algorithm Adam (adaptive moment estimation) is used. The mini-batch size is chosen to be 1, so the model provides the lowest generalization error, but has to be stabilized with a low learning rate of $2 \cdot 10^{-4}$ (Goodfellow, 2016, p. 279). The model is trained over 15 epochs. After extracting the weights of the embeddings and replacing the levels of the categorical variables with the respective weight vectors, an OLS regression is conducted on the encoded data to make the predictions.

6.3.4 Group Lasso

The group lasso is implemented with the R-package *gglasso* by Yang & Zou (2015). *gglasso* computes the solution path for a given group-lasso penalized problem with a unified algorithm called groupwise-majorization-descent (p. 1129). This algorithm does not require the covariates in the design matrix to be group-wise orthonormal in contrast to the established *grplasso*-package which uses classical groupwise descent (p. 1129). That is one advantage of *gglasso*, since the groupwise orthonormal condition would get violated when a fraction of the observation is removed during cross-validation (pp. 1132–1233). The second advantage is speed, as the descent algorithm of *gglasso* is 10 to 15 times faster compared to the one from *grplasso* (p. 1140). For the prediction task, the optimal value of the penalty parameter λ is found by a 5-fold cross-validation given the 1-standard deviation rule (Bilder & Loughin, 2015, p. 278).

⁹ https://github.com/compstat-lmu/paper_2021_categorical_feature_encodings

6.3.5 Sparse-Group Lasso

The R-package *seagull* by Klosa, Simon, Westermarck, Liebscher & Wittenburg (2020) is used for the implementation of the sparse-group lasso. The function applies proximal gradient descent to solve the optimization problem (p. 2). Also thanks to warm starts, *seagull* is 20 times faster than the established R package *SGL*, whereas the results are very similar (p. 2). By setting $\alpha = 1$, *seagull* could also solve the group lasso, but is slower than *gglasso*. In the prediction task, a grid search for the optimal mixing parameter α is done via cross-validation. As suggested by Simon et al. (2013), the grid includes the values 0.05 and 0.95 (p. 237). Once α is known, a second 5-fold cross-validation is performed to additionally determine the optimal λ , that is used for the prediction.

6.3.6 Fused Lasso

For the implementation of the fused lasso, the R-package *genlasso* is chosen. As suggested in the manual, the wrapper function for the one-dimensional fused lasso is directly applied. The penalty parameter λ for the prediction is then found by 5-fold cross-validation. As predictions with certain datasets explode (values $>10^{30}$) when choosing a lambda below a certain threshold for inexplicable reasons, a comparison with the predictions from the cross-validation follows afterwards. If the predictions are indeed that high, the lambda is increased stepwise by 10% until a reasonable result is obtained.

6.4 Results

To evaluate and compare the prediction accuracy of each method, the mean-squared error (MSE) is used as performance measure. Its formula can be decomposed into the sum of the squared bias and the variance of the output; hence it takes both measures into account. The MSE is the squared difference of the true output and the estimated output, averaged over the whole sample:

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 = \frac{1}{n} \|y - \hat{y}\|_2^2 \quad (21)$$

6.4.1 Simulation

The simulation results averaged over 1000 runs are shown below:

	OLS	Lasso	LR	EE	GLMM	GL	SGL	FL	SL
N = 300 Cat: 4 Cont: 4 Levels: 8, 5, 4, 3									
DGP 1	3.066 (1)	3.385 (1.104)	3.27 (1.066)	3.168 (1.033)	3.369 (1.099)	3.103 (1.012)	3.111 (1.015)	2.911 (0.949)	2.877 (0.938)
DGP 2	4.639 (1)	5.032 (1.085)	5.144 (1.109)	4.959 (1.069)	5.252 (1.132)	4.653 (1.003)	4.692 (1.011)	4.676 (1.008)	4.53 (0.977)
DGP 3	3.869 (1)	4.186 (1.082)	6.092 (1.574)	4.948 (1.279)	5.448 (1.408)	3.874 (1.001)	3.904 (1.009)	3.882 (1.003)	3.852 (0.996)
DGP 4	3.148 (1)	3.443 (1.094)	5.457 (1.733)	4.041 (1.284)	4.415 (1.402)	3.145 (0.999)	3.272 (1.039)	3.213 (1.021)	3.176 (1.009)
N = 300 Cat: 4 Cont: 4 Levels: 30, 15, 10, 5									
DGP 1	3.67 (1)	3.423 (0.933)	3.037 (0.828)	3.098 (0.844)	3.091 (0.842)	3.064 (0.835)	3.599 (0.981)	2.758 (0.752)	2.757 (0.751)
DGP 2	5.401 (1)	5.499 (1.018)	4.737 (0.877)	4.855 (0.899)	4.74 (0.878)	4.863 (0.9)	5.215 (0.966)	4.427 (0.82)	4.325 (0.801)
DGP 3	5.136 (1)	5.429 (1.057)	7.548 (1.47)	6.833 (1.33)	6.531 (1.272)	4.858 (0.946)	5.048 (0.983)	4.872 (0.949)	4.732 (0.921)
DGP 4	4.342 (1)	4.616 (1.063)	6.682 (1.539)	6.044 (1.392)	5.61 (1.292)	4.072 (0.938)	4.213 (0.97)	4.213 (0.97)	4.097 (0.944)
N=300 Cat: 1 Cont: 7 Levels: 80									
DGP 1	9.397 (1)	7.751 (0.825)	6.887 (0.733)	6.977 (0.742)	6.841 (0.728)	6.816 (0.725)	8.479 (0.902)	6.756 (0.719)	6.764 (0.72)
N = 300 Cat: 20 Cont: 0 Levels: 3, 3, ..., 3									
DGP 1	1.857 (1)	2.018 (1.087)	3.764 (2.028)	1.857 (1)	3.812 (2.053)	1.917 (1.033)	1.847 (0.995)	1.538 (0.829)	1.539 (0.829)

This table contains the average mean-squared error over 1000 runs of each method for every data-generating process. The numbers in brackets are the values relative to the OLS. Notation: LR = low-rank encoding, EE = entity embedding, GL = group lasso, SGL = sparse-group lasso and FL = fused lasso.

Table 7: Results of the simulations

6.4.2 Real Data

In order to prevent a disadvantageous random split when dividing the dataset into a training set and a test set, the average MSE from 5 runs was taken. The results are presented in Table 8:

	OLS	Lasso	LR	EE	GLMM	GL	SGL	FL	SL
Avocado	0.067 (1)	0.069 (1.018)	0.094 (1.389)	0.089 (1.327)	0.068 (1.003)	0.154 (2.283)	2.067 (30.69)	0.121 (1.796)	0.068 (1.003)
Ames Housing	1236 (1)	1069 (0.864)	1259 (1.018)	1188 (0.961)	1039 (0.841)	1968 (1.592)	2552 (2.064)	927 (0.75)	983 (0.795)
Employee salaries	112.3 (1)	123.3 (1.098)	531.4 (4.731)	456.7 (4.066)	137.1 (1.22)	437.1 (3.891)	869.9 (7.744)	150.1 (1.336)	128.7 (1.143)
Student Perform.	7.946 (1)	7.935 (0.999)	7.434 (0.936)	7.264 (0.914)	7.631 (0.96)	7.017 (0.883)	7.402 (0.932)	7.123 (0.897)	6.966 (0.877)

This table contains the mean-squared errors for each method on real datasets. The numbers in brackets are the values relative to the OLS. The MSE for Ames housing and employee-salaries was divided by 1'000'000.

Table 8: Prediction accuracy on real datasets

6.5 Discussion

Initially, the results of the simulations are evaluated in general, then they are compared with those of the real data, and finally the methods are analyzed separately. The MSE relative to the OLS forms the basis for the discussion.

Each method improved from scenario 1 to scenario 2 as the number of feature levels increased. However, the encoding methods were found to be less stable with respect to the variability of the regression coefficients (DGP 2-4). This finding is surprising, since the lasso type algorithms were expected to have more difficulties with variability within groups of dummy variables according to Huang & Montoya (2020, p. 23). An explanation for this could lie in the nature of the encoding methods. Since the encodings are partly derived from information from other variables, they may become less accurate if some of the variables have little explanatory power, or in other words have regression coefficients that are close to zero.

The extreme case with only one high-cardinality categorical variable was simulated with uniform coefficients. Thus, all methods obtained exceptional results and a

significant improvement from the main scenarios. In the other extreme case with many categorical variables but only a few levels, the lasso types performed better than the OLS, while the Encoding methods did not prove to be effective. In the following, we examine whether the conclusions also apply to the real data.

For the *avocado* dataset, none of the methods except GLMM could compete with the OLS and lasso. One reason for this could be that its categorical variable might have many levels, but not compared to the sample size. Thus, the OLS can handle them without any problems (Pargent et al. 2021, p. 2). The sparse-group lasso struggles a lot with that dataset. Any attempt to optimize the tuning of the hyperparameters does not bring any improvement.

The *ames housing* set, consisting mainly of categorical variables, yielded mixed results. While the fused lasso and GLMM perform well, the group lasso and again the sparse-group lasso lag behind.

In the *employee salaries* data, 3 of 6 predictors are categorical variables with a high number of feature levels. Hence, according to the simulation results, it should be expected, that the methods perform well. On the contrary, none of the methods were as good as the OLS or lasso. Apart from GLMM and fused lasso, the methods performed insufficiently. One reason for this could be that the feature levels of the high-cardinal predictor “positions” have a highly variable impact on the output like with DGP 3-4.

With the *student performance* data, each method outperformed the OLS and the lasso. The fact, that half of the predictors are categorical with a large number of levels relative to the sample size certainly played a role, similar to Ames Housing. Notably, the entropy across levels is relatively small for both datasets, in contrast to employee salaries.

Besides the difficulties with DGP 2-4 for the encoding methods and the strong performance of the fused lasso, no method stood out in the simulation. With real data, however, some methods encountered issues. In the following, those are examined again in more detail.

Given that the *low rank encoding* only takes care of the categorical variable with the most feature levels and uses label encoding for the other predictors, the method was expected to perform worse compared to the other methods. The student performance data was supportive for the other integer encoded columns, as it mainly included ordered categorical variables.

Although the neural network used to produce the *entity embeddings* was not tuned before applying on real data, the performance was in line with the other methods. With a better understanding of neural networks, one could possibly achieve far better results (Hastie et al., 2009, pp. 397–401).

The regularized target encoding approach using *GLMM* produced the best results on average with real data compared with the other presented methods. In Pargent et al. (2021), this approach also worked best for employee salaries data, but there the encoded data was feed into several other methods rather than OLS (p. 18).

The different DGP's were mainly set up that way because of the findings of Huang & Montoya (2020), which were that the *group lasso* leads to overfitting and worse performance by including categorical variables more likely, when few dummies have a higher coefficient than the mean of its group (p. 24). However, the relative MSE's of the different DGP's were quite stable, which could be because the chosen variability was too low when selecting the beta coefficients. In the simulations of Yuan & Lin (2006) the test error of the group lasso was always lower than the one of OLS (p. 61). In contrast to our simulations, their data consisted of several uninformative predictors (p. 60). In the experiment with real data, they only compared group methods, but not with the OLS (p. 64).

The *sparse-group lasso* performed quite alright on the simulated data and was the most robust in DGP 2-4. This comes from the fact that it uses $\alpha = 0.05$ when expecting group-wise sparsity, and $\alpha = 0.95$ when expecting overall sparsity, for which our DGP's also accounted (Simon et al., 2013, p. 237). Nevertheless, the method was clearly inferior on real data. Simulations in Simon et al. (2013) and Klosa et al. (2020) imply that the method is more appropriate for high dimensional data like samples of DNA or protein spectroscopy with few samples and thousands of predictors. There, the sparse-group lasso successfully excludes the uninformative features and yields good results (p. 243; p. 2). Since the MSE of the sparse-group lasso diverged much from the normal lasso even in a test with $\alpha = 1$ on real data with all possible combinations of hyperparameters, allows doubting the other results of the experiment and the choice of the R package to implement the algorithm.

The *fused lasso* is the most convincing method tested in the experiment. The algorithm provides strong results with many categorical variables as well as with large numbers of feature level).

The categorical *super learner* outperformed OLS and the lasso in almost every simulation. However, it did not always overperform the best included method, namely, the group lasso or the fused lasso. One reason for this could be the small sample size, which might distort the cross-validation results, leading to inaccurate weights of the estimators.

In summary, the performance of all tested methods improves as the number of levels of categorical variables increases. Above a certain threshold, also the encoding methods outperform the OLS and compare favorably to the lasso when there is not too much variability within the coefficients of the categorical variables. This behavior is demonstrated in more detail in Figure 7:

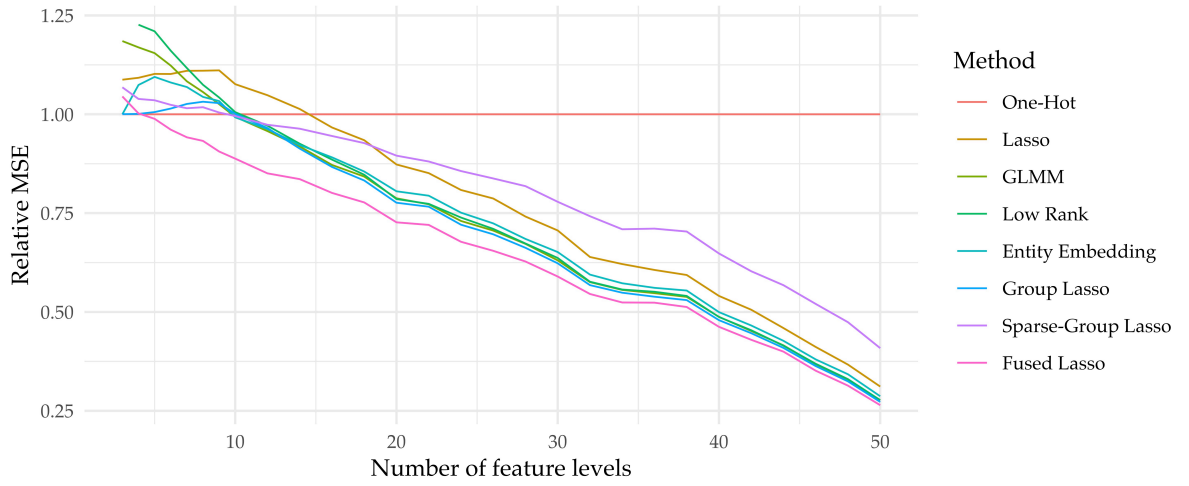


Figure 7: Relative performance of the methods depending on the number of feature levels. Simulation with 250 runs for each data point with DGP 1. The data consists of 3 continuous and 3 categorical variables, and the number of levels on the x-axis corresponds to each categorical variable. The y-axis denotes the MSE relative to the OLS.

Again, the fused lasso delivers consistently strong results, whereas the sparse-group lasso is the only one that lags behind the lasso. Thus, all methods except the one just mentioned perform better than the lasso on data with many feature levels. However, with real data, some of our methods outperformed the OLS in only 2 of 4 cases, which was not expected given that some datasets like the employee salaries contain highly cardinal categorical variables. Due to the scope of this thesis, the reasons for that matter could not be elaborated completely.

7 Conclusion

This thesis aimed to raise the awareness of methods for dealing with categorical predictors that take group information into account, which would be wasted by using one-hot encoding. For this purpose, encoding methods that represent categorical predictors in a lossless manner as well as lasso-type algorithms that produce a sparse solution were presented.

Applying these methods to data containing only few categorical variables and feature levels proved to be less effective, as shown by the simulation, where the OLS always produced the best results. Nevertheless, when the cardinality of the data compared to the sample size becomes larger, these methods should not be avoided. According to the simulation with high-cardinality features, all methods outperformed not only the OLS but also the lasso. However, when applied to real data of that type, mixed results were obtained, although revealing the more robust methods. After all, only the fused Lasso and GLMM encoding produced stable, but not always superior results across all datasets, while the ensemble learner mostly outperformed the best included method. Therefore, the author suggests considering GLMM to encode high-cardinality categorical data or applying the fused lasso. Besides, the fused lasso is well suited to be included into an ensemble learner due to its simple implementation.

This study may suffer from some limitations. Since the results of the simulations diverge considerably from those with real data, it suggests that the data-generating process was not realistic enough. Moreover, encoding methods were systematically evaluated with the OLS even when there was evidence of a non-linear relationship with the output.

Further research could address this aspect and combine the encoding methods with other algorithms than OLS. In addition, it could be determined what properties high-cardinal data should additionally satisfy for the methods to work best, since they performed poorly on the employee salaries dataset even though it has such a structure. Finally, although entity embedding did not perform best in this study, it could be more powerful with a better understanding of neural nets. Therefore, more literature on the effect of the structure of a neural net on entity embedding would be useful.

References

- Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3(6), 1137-1155.
- Bilder, C. R., & Loughin, T. M. (2015). *Analysis of Categorical Data with R*. Boca Raton: CRC Press.
- Breiman, L. (1996). Stacked Regressions. *Machine Learning*, 24(1), 49-64.
- Choi, Y., Park, R., & Seo, M. (2012). *Lasso on Categorical Data*. Retrieved from <http://cs229.stanford.edu/proj2012/ChoiParkSeo-LassoInCategoricalData.pdf>
- Cortez, P., & Silva, A. (2008). Using Data Mining to Predict Secondary School Student Performance. In A. Brito, & J. Teixeira, *Proceedings of 5th Future Business Technology Conference (FUBUTEC 2008)* (pp. 5-12). Porto.
- De Cock, D. (2011). Ames, iowa: Alternative to the boston housing data as an end of semester regression project. *Journal of Statistics Education*, 19(3).
- Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, G. K., & Harshman, R. (1999). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6), 391-407.
- Fisher, W. D. (1958). On Grouping for Maximum Homogeneity. *Journal of the American Statistical Association*, 53, 789-798.
- Forlenza, V., Lapiello, E., & Fowler, A. (2019). *A Better Way? Forecasting with Embeddings*. Retrieved from <https://medium.com/bcggamma/a-better-way-forecasting-with-embeddings-8f45e7065f2b>
- Gertheiss, J., & Tutz, G. (2010). Sparse Modelling of Categorical Explanatory Variables. *The Annals of Statistics*, 4(4), 2150-2180.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. The MIT Press.
- Guo, C., & Berkhahn, F. (2016). Entity Embeddings of Categorical Variables. Retrieved from <https://arxiv.org/pdf/1604.06737.pdf>
- Hand, D. J., & Henley, W. E. (1997). Statistical classification methods in consumer credit scoring: A review. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 160, 523-541. Retrieved from <https://doi.org/10.1111/j.1467-985X.1997.00078.x>

- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. New York: Springer.
- Heiler, P., & Mareckova, J. (2021). Shrinkage for Categorical Predictors. *Journal of Econometrics*, 223(1), 161-189. Retrieved from <https://doi.org/10.1016/j.jeconom.2020.07.051>
- Huang, Y., & Montoya, A. (2020). *Lack of Robustness of Lasso and Group Lasso with Categorical Predictors: Impact of Coding*. Retrieved from <https://escholarship.org/uc/item/40b200z6>
- Johannemann, J., Hadad, V., Athey, S., & Wager, S. (2019). *Sufficient Representation of Categorical Variables*. Retrieved from <https://arxiv.org/pdf/1908.09874.pdf>
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., & Te, Q. (2017). LightGBM: A Highly Efficient Gradient Boosting Tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett, *Advances in Neural Information Processing Systems 30 (NIPS 2017)* (pp. 3149-3157). MIT Press.
- Kiggins, J. (2018, May). *Avocado Prices*. Retrieved August 2021, from Kaggle: <https://www.kaggle.com/neuromusic/avocado-prices/version/1>
- Kim, J. (2014). Comparison of Lasso Type Estimators for High-Dimensional Data. *Communications for Statistical Applications and Methods*, 21(4), 348-361.
- Klosa, J., Simon, N., Westermarck, P. O., Liebscher, V., & Wittenburg, D. (2020). seagull: lasso, group lasso and sparse-group lasso regularisation for linear regression models via proximal gradient descent. *BMC Bioinformatics*.
- Kuhn, M., & Johnson, K. (2019). *Feature Engineering and Selection: A Practical Approach for Predictive Models*. Retrieved from <http://www.feat.engineering/index.html>
- LightGBM. (n.d.). Retrieved August 2021, from <https://lightgbm.readthedocs.io/en/latest/Features.html#optimal-split-for-categorical-features>
- Mezzogori, D., & Zammori, F. (2019). An Entity Embeddings Deep Learning Approach for Demand Forecasting of Highly Differentiated Products. *Procedia Manufacturing*, 39, 1793-1800. Retrieved from <https://doi.org/10.1016/j.promfg.2020.01.260>
- Micci-Bareca, D. (2001). A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *SIGKDD Exploration Newsletter*, 3(1), 27-32. Retrieved from <https://doi.org/10.1145/507533.507538>

- Naimi, A. I., & Balzer, L. B. (2018). Stacked Generalization: an introduction to super learning. *European Journal of Epidemiology*, 33(5), 459-464.
- Pargent, F. (2019, January 27). *OpenML*. Retrieved August 2021, from <https://www.openml.org/d/41445>
- Pargent, F., Pfisterer, F., Janek, T., & Bischl, B. (2021). *Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features*. Retrieved from <https://arxiv.org/pdf/2104.00629.pdf>
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*(323), 533-536.
- Simon, N., & Tibshirani, R. (2012). Standardization and the Group Lasso Penalty. *Statistica Sinica*, 22(3), 983-1001.
- Simon, N., Friedman, J., Hastie, T., & Tibshirani, R. (2013). A Sparse-Group Lasso. *Journal of Computational and Graphical Statistics*, 22(2), 231-245.
- Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 58(1), 267-288.
- Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., & Knight, K. (2005). Sparsity and Smoothness via the Fused Lasso. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(1), 91-108.
- Tutz, G. (2011). *Regression for Categorical Data*. New York: Cambridge University Press.
- van der Laan, M. J., Polley, E. C., & Hubbart, A. E. (2007). Super Learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 1309-1327.
- West, B. T., Welch, K. B., & Galecki, A. T. (2014). *Linear Mixed Models: A Practical Guide Using Statistical Software*. Boca Raton: CRC Press.
- Wolpert, D. H. (1992). Stacked Generalization. *Neural Networks*, 5, 241-259.
- Yang, Y., & Zou, H. (2015). A Fast Unified Algorithm for Solving Group-Lasso Penalized Learning Problems. *Statistics and Computing*, 25, 1129-1141.
- Yuan, M., & Lin, Y. (2006). Model Selection and Estimation in Regression with Grouped Variables. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 68(1), 49-67.
- Zou, H., & Hastie, T. (2005). Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 76(2), 301-320.

Appendix

Appendix A: R-Code

The R-Code is stored in a Github-Repository accessible with the following URL:

<https://github.com/dabrunner/master-thesis>

Appendix B: Simulations in Detail

Cont	Cat	Levels	Beta coefficients	Variance among groups:
4	4	8, 5, 4, 3	$\beta = \{1, \underbrace{1,1,1,1,1,1,1}_{\mu_2=1, var_2=0}, \underbrace{1,1,1,1}_{\mu_3=1, var_3=0}, \underbrace{1,1,1}_{\mu_4=1, var_4=0}, \underbrace{1,1,1,1,1}_{\mu_5=1, var_5=0}\}$	$Var(\mu_{1,...,9}) = 0$
4	4	8, 5, 4, 3	$\beta = \{1.1, \underbrace{0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9}_{\mu_2=0.9, var_2=0}, \underbrace{0.7, 0.7, 0.7, 0.7, 2.7, 2.7, 2.7}_{\mu_3=0.7, var_3=0}, \underbrace{0.9, 0.9}_{\mu_4=2.7, var_4=0}, \underbrace{-0.6, 0.9, 1.0, 2.6}_{\mu_5=0.9, var_5=0}\}$	$Var(\mu_{1,...,9}) = 0.998$
4	4	8, 5, 4, 3	$\beta = \{1, \underbrace{2.5, 0.27, 0.6, -0.45, 1.13, 0.98, 1.97}_{\mu_2=1, var_2=1}, \underbrace{-0.29, 0.67, 1.75, 1.87}_{\mu_3=1, var_3=1.03}, \underbrace{1.99, 0, 1.02}_{\mu_4=1, var_4=0.99}, \underbrace{1.71, 0.29, 1, 1, 1, 1}_{\mu_5=1, var_5=1.01}\}$	$Var(\mu_{1,...,9}) = 0$
4	4	8, 5, 4, 3	$\beta = \{1.1, \underbrace{-1.54, -2.32, -0.92, -0.49, 0.63, 0.24, -0.6}_{\mu_2=-0.714, var_2=1.014}, \underbrace{1.42, 0.96, 0.88, 0.43}_{\mu_3=-0.443, var_3=1.038}, \underbrace{0.15, -1.84, -1.16}_{\mu_4=-0.95, var_4=1.023}, \underbrace{1.42, -0.02, -0.1, -1.2, -1.9, 0.4}_{\mu_5=0.7, var_5=1.037}\}$	$Var(\mu_{1,...,9}) = 0.999$
4	4	30, 15, 10, 5	$\beta = \{1, \underbrace{1, \dots, 1}_{\mu_2=1, var_2=0}, \underbrace{1, \dots, 1}_{\mu_3=1, var_3=0}, \underbrace{1, \dots, 1}_{\mu_4=1, var_4=0}, \underbrace{1, 1, 1, 1, 1}_{\mu_5=1, var_5=0}\}$	$Var(\mu_{1,...,9}) = 0$
4	4	30, 15, 10, 5	$\beta = \{1.1, \underbrace{0.9, \dots, 0.9}_{\mu_2=-0.9, var_2=0}, \underbrace{0.7, \dots, 0.7}_{\mu_3=0.7, var_3=0}, \underbrace{2.7, \dots, 2.7}_{\mu_4=2.7, var_4=0}, \underbrace{0.9, \dots, 0.9}_{\mu_5=0.9, var_5=0}, -0.6, 0.9, 1, 2.6\}$	$Var(\mu_{1,...,9}) = 0.998$
4	4	30, 15, 10, 5	$\beta = \{1, \underbrace{1.03, 1.04, 2.4, 2.2, 1.6, 1.33, 0.78, 0.6, 1.2, 1.95, 0.86, 0.3, 1.1, 0.4, -1.27, 1.9, 0.36, -0.37, 0.87}_{\mu_2=1, var_2=1.01}, \underbrace{-0.49, 1.64, 1.4, 1.75, 1.29, 0.3, 1.01, 1.61, 1.85, 1.04, -0.52, 0.17, 0.65, 2.91}_{\mu_3=1.001, var_3=0.999}, \underbrace{1.56, 0.74, 0.41, 1.15, 1.5, 2.23, 2.2, -0.11, -0.68}_{\mu_3=1, var_3=1.001}, \underbrace{2.5, 0.6, 0.6, 0.33}_{\mu_5=1.008, var_5=1.006}, 1, 1, 1, 1\}$	$Var(\mu_{1,...,9}) = 0$
4	4	30, 15, 10, 5	$\beta = \{1.1, \underbrace{2.2, -2, -1.4, 0.75, -0.73, 1.15, -1.7, -1.36, -0.5, -0.9, 0.78, -0.43, -0.8, 0, 0.33, 0.2}_{\mu_2=-0.713, var_2=1}, \underbrace{-0.6, 1.4, 1.1, -1.2, 0.9, 0.4, -0.8, 0.8, 0.5, 1.2}_{\mu_3=0.443, var_3=1}, \underbrace{-1.7, -0.1, -1.9, 0.4, -1.3, -1.1, 0.54, -2.2, -1.2}_{\mu_3=0.951, var_3=1}, \underbrace{-0.24, 1.5, -0.09, 1.65}_{\mu_5=0.703, var_5=1.01}, -0.1, -1.2, -1.9, 0.4\}$	$Var(\mu_{1,...,9}) = 1$
0	20	3, 3, ...	$\beta = \{1, \underbrace{1,1}_{\mu_2=1, var_2=0}, \underbrace{1,1}_{\mu_3=1, var_3=0}, \underbrace{1,1}_{\mu_4=1, var_4=0}, \underbrace{1,1}_{\mu_5=1, var_5=0}, \dots, \underbrace{1,1}_{\mu_{61}=1, var_{61}=0}\}$	$Var(\mu_{1,...,21}) = 0$
1	7	80	$\beta = \{1, \underbrace{1, \dots, 1}_{\mu_2=1, var_2=0}, 1, 1, 1, 1, 1, 1, 1\}$	$Var(\mu_{1,...,9}) = 0$

Declaration of authorship

"I hereby declare

that I have written this thesis without any help from others and without the use of documents and aids other than those stated above;

that I have mentioned all the sources used and that I have cited them correctly according to established academic citation rules;

that I have acquired any immaterial rights to materials I may have used such as images or graphs, or that I have produced such materials myself;

that the topic or parts of it are not already the object of any work or examination of another course unless this has been explicitly agreed on with the faculty member in advance and is referred to in the thesis;

that I will not pass on copies of this work to third parties or publish them without the University's written consent if a direct connection can be established with the University of St. Gallen or its faculty members;

that I am aware that my work can be electronically checked for plagiarism and that I hereby grant the University of St. Gallen copyright in accordance with the Examination Regulations in so far as this is required for administrative action;

that I am aware that the University will prosecute any infringement of this declaration of authorship and, in particular, the employment of a ghostwriter, and that any such infringement may result in disciplinary and criminal consequences which may result in my expulsion from the University or my being stripped of my degree."

St. Gallen, August 23, 2021

Daniel Brunner

By submitting this academic term paper, I confirm through my conclusive action that I am submitting the Declaration of Authorship, that I have read and understood it, and that it is true.