A TREE BASED CATEGORICAL VARIABLE ENCODING STRATEGY IN SUPERVISED LEARNING TASKS

by

Mine Gazioğlu

B.S., Molecular Biology and Genetics, Boğaziçi University, 2018

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computational Science and Engineering
Boğaziçi University

2022

ACKNOWLEDGEMENTS

I am excited to complete my master's thesis, which I have prepared with serious effort and devotion over the years. I will use this section as an opportunity to thank the people who have helped and encouraged me to complete this thesis. First of all, I would like to thank my thesis supervisor, Mustafa Gökçe Baydoğan, who guided me in my studies, encouraged me at all times and given me invaluable feedbacks. I would like to thank Emel Şeyma Küçücaşcı and Bertan Badur, who participated in my thesis defense and provided constructive criticism at critical points to take the final shape of the thesis. Finally, I would like to express gratitude to my family who has been there for me throughout my life.

ABSTRACT

A TREE BASED CATEGORICAL VARIABLE ENCODING STRATEGY IN SUPERVISED LEARNING TASKS

Categorical variables are present in most real-world datasets, often consisting of a high number of levels, referred to as high-cardinality categorical variables. Most machine learning algorithms do not have an innate mechanism to deal with categorical variables, hence, their encoding is necessary. Categorical variable encoding is the general term for the conversion of nominal independent variables to a numerical format. Many encoding strategies exist, and they are discussed in this thesis. This thesis presents a novel encoding strategy, categorical split encoding, and also provides an analysis of existing encoding methods. Categorical split encoding uses primary and surrogate split information as the vector representation for categorical variables, through a tree-based algorithm, this method outputs binary columns for each categorical variable making use of target information. Missing values are imputed by using surrogate information, while clustering similar values together based on the path they take through the decision tree algorithm. Various existing encoding strategies are benchmarked for comparison with the proposed strategy. The performance of categorical split encoding and other encoding methods is compared with three different machine learning algorithms (generalized linear models, random forest and xgboost) using datasets from regression, binary and multiclass classification settings. Datasets used are made publicly available for replication purposes. As a result, categorical split encoding provides competitive results compared to existing encoding strategies in various datasets.

ÖZET

AĞAÇ BAZLI METOD İLE KATEGORİK DEĞİŞKENLERİN SAYISALLAŞTIRILMASI

Kategorik değişkenler çoğu datasette bulunur, genellikle yüksek kardinalite kategorik değişkenler olarak adlandırılır ve çok sayıda seviyeden oluşurlar. Çoğu makine öğrenme algoritması, kategorik değişkenleri numerik formata dönüştürmek için bir mekanizmaya sahip değildir, bu nedenle kategorik değişkenlerin kodlanmaları gereklidir. Kategorik değişkenlerin kodlanması, nominal değişkenlerin numerik formata çevrilmesi için kullanılan genel terimdir. Kategorik değişkenlerin kodlanması için birçok kodlama stratejisi mevcuttur ve bu stratejiler bu tezde incelenmektedir. Bu tez, yeni bir kodlama stratejisi olan kategorik bölünmüş kodlamayı ve var olan kodlama metodlarının analizini sunmaktadır. Kategorik bölünmüş kodlama, ağaç tabanlı bir algoritma aracılığıyla, kategorik değişkenler için vektör temsili olarak birincil ve yedek bölünmüş bilgileri kullanır; bu yöntem, bağımlı değişken bilgilerini kullanarak her kategorik değişken için ikili kolonlar üretir. Eksik değerler, karar ağacı algoritmasında izledikleri yola göre benzer kategoriler bir araya kümelenirken, yedek bölünmüş bilgiler kullanılarak doldurulur. Kategorik bölünmüş kodlama ve var olan kodlama metodları karşılaştırılmaktadır. Kategorik bölünmüş kodlama ve diğer kodlama yöntemlerinin performansı, regresyon, ikili ve çok sınıflı sınıflandırma ayarlarından veri kümeleri kullanılarak üç farklı makine öğrenme algoritması (genelleştirilmiş doğrusal modeller, rastgele orman ve xgboost) ile karşılaştırılmıştır. Kullanılan datasetler, deneylerin tekrarlanabilmesi amacıyla herkese açık hale getirilmiştir. Sonuç olarak, kategorik bölünmüş kodlama, mevcut kodlama stratejilerine kıyasla rekabetçi sonuçlar sağlar.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

b_i	A level of a categorical variable, $i = 1, \ldots, K$
C	Number of categorical columns
P	Number of numerical columns
$i(t)$	Impurity function at any node t
J	Number of classes for a classification problem
K	Number of levels in a categorical variable
M	Number of independent variables
N	Number of observations
N_j	Number of samples in class j
$N_j(t)$	Number of samples in class j going to node t
p_R	Percentage of instances sent to t_R
p_L	Percentage of instances sent to t_L
$p(.,.)$	Joint probability
$p(.\|.)$	Conditional probability
$\boldsymbol{Q}$	Set of binary questions for any independent variable X
s	Split of any node t
t	Any node of a decision tree
t_L	Left child node of a node t
t_R	Right child node of a node t
$\boldsymbol{X}$	Feature space
X_i	Independent variable, $i = 1, \ldots, M$
y	Target variable
ΔI	Goodness of split function
Δ^*	Best split at any node t
$\tilde{\Delta}$	Best surrogate split at any node t
$\pi(j)$	Prior probability of class j

LIST OF ACRONYMS/ABBREVIATIONS

CART	Classification and regression trees
CSE	Categorical split encoding
CV	Cross-validation
Freq	Frequency encoding
Int	Integer encoding
Leaf	Leaf encoding
LR	Generalized linear models for classification and regression
MR	Missing rate
OHE	One-hot encoding
RF	Random forest
XGB	XGBoost

1. INTRODUCTION

Categorical variables are present in most real-world datasets, whereas a numerical feature matrix is required for the majority of machine learning algorithms, as numerical operations are not valid for categorical variables. Only certain operations are defined while working with categorical variables, such as equivalence, counting and finding the mode. Although, some algorithms can handle categorical variables, such as decision trees, decision tree ensemble methods and Naive Bayes, some implementations of these methods cannot deal with categorical variables [1]. Consequently, when categorical variables exist in the feature matrix, there is a need for encoding so that they are transformed into a numerical vector representation before fitting statistical models.

Categorical variables, such as gender, state and job title, are discrete entities that can be from a set of limited possible values, where each one of the possible values is referred to as a level or a category. These discrete entities are common in tabular datasets and the quality of the resulting numerical representation obtained by encoding them is crucial for the performance of the statistical model used. Cardinality of a categorical variable is the number of distinct categories it consists of [2,3]. Categorical variables consisting of a large number of distinct categories are referred to as high-cardinality categorical variables. The lower threshold for a categorical variable to be considered as high-cardinality is often determined by choice. For instance, the lower threshold can be set as 100 [4].

Categorical variables can broadly be classified into two categories: nominal and ordinal. Nominal categorical variables include discrete categories with no meaningful order, whereas for ordinal categorical variables, categories can be ordered in a sensible way, but this order is not yet numerically quantified. Hence, ordinal categorical variables can be encoded by assigning numerical values in the natural ordering of the variable, while with nominal categorical variables this method would not make sense. Categorical variables can also be present as text in datasets. For text categorical vari-

ables, such as employee position titles and customer surveys, use of similarity measures has been studied in numerous papers. For instance, if a categorical variable consists of employee position titles, such as "Police Aide", "Police Officer" and "Master Police Officer", it can observed that there are common words and subwords in the given levels. Using these similarities in levels, an encoding scheme is performed. However, this subject is out of scope for this thesis and readers are referred to referenced papers for a comprehensive review [5–7].

There have been benchmarking studies conducted to compare encoding strategies for which type of dataset they would be the best fit [8–12]. The efficiency, namely whether an encoding strategy will perform well, is dependent on the dataset and the nature of the categorical variable. For instance, if the cardinality is not high and the levels are independent, meaning each level is not syntactically or semantically related, then, one-hot encoding, which creates a binary column for each level, can be used as the encoding strategy. Conversely, if the cardinality is high, use of one-hot encoding will lead to a high-dimensional vector space, causing computational complexity. In such a case, another strategy to reduce the cardinality, such as categorical clustering, can be performed [5, 8, 13]. Tree based methods for categorical variable encoding are efficient, as classification and regression trees (CART) can naturally handle categorical variables [14]. There are other decision tree algorithms such as CHAID [15]. CHAID is prone to be used with categorical targets and unlike CART, it splits in a multiway fashion. When using CART, dissimilar to one-hot encoding, proximity between levels of a categorical variable is implicitly provided by a decision tree. Decision trees also have surrogate split mechanism that can innately handle missing values in train and test sets without imputing [8, 14, 16].

This thesis proposes a new categorical variable encoding method, categorical split encoding, and also evaluates numerous existing encoding methods in comparison to categorical split encoding. The differences in various encoding strategies in terms of performance and robustness of the method when the percentage of missing values increases on datasets from classification and regression settings, containing categorical

variables of high, medium and low-cardinality nature, is investigated. The problem of high-dimensionality generated by performing one-hot encoding on datasets with large cardinality is mitigated by clustering the levels that go through a similar path along the tree with categorical split encoding. Also, one-hot encoding assumes each level to be equally distant to one another, when in many cases some levels are more similar to one another than others [5, 17]. This notion of proximity is supplied by the tree structure, hence, with categorical split encoding, levels that are more similar, have similar representations. Frequency encoding, in a way clusters levels based on how frequent they are. This only works when there is a functional relationship between the frequency of the levels and the target variable [18]. Label encoding assigns integers to levels by lexical ordering, however, lexical ordering, in most cases, does not imply the actual relationship between the levels. Methods like hierarchical clustering, do not take into consideration the relation with the target variable [19]. Tree based methods proposed, such as leaf encoding, only considers the relationship between one categorical variable and the target variable, one at a time, without taking into account, continuous variables and other categorical variables [20]. Categorical split encoding uses the relation between the target variable and the categorical variable. Also, implicity uses the effect of other variables, whether continuous or categorical. Missing values problem, in most encoding methods, is solved by imputing the dataset before fitting a machine learning model with an imputation strategy. Categorical split encoding imputes the categorical variables with surrogate split information of the tree, without the need to impute after encoding.

The organization of the thesis is as follows: Chapter 2 provides literature review of the encoding strategies. Chapter 3 introduces theoretical background to grasp the details of the subjects that are mentioned in the rest of this thesis. Tree structure and operating principles of classification and regression trees (CART) and the concept of surrogate splits are introduced. Chapter 4 provides detailed description and working mechanism of categorical split encoding. Pseudocode along with text explanation is supplied. Chapter 5 outlines the design of the experiments and in Chapter 6, results of the experiments are presented and discussed. Finally, Chapter 7 concludes the thesis.

2. LITERATURE REVIEW

One of the most common encoding strategies when dealing with categorical variables is one-hot encoding (OHE), which yields a vector space in which each level is equidistant and orthogonal to each other. For each level in a categorical feature, a new binary feature is added indicating it. For instance, let there be a categorical variable that consists of unique levels that form the set {"*blue*", "*green*", "*red*"}. Each category can be encoded respectively as $\{[1,0,0],[0,1,0],[0,0,1]\}$. For this categorical variable, since there are three unique levels, three binary columns are generated, each column corresponding to a level of the original variable. Heuristically, this method implies that categories are assumed to be unrelated syntactically or semantically, when in many cases categorical levels might be related. Then, the levels that are related should be closer in the feature space. This is one of the method's disadvantages as categories are not always equidistant to one another in feature space. Another disadvantage of the method is observed as the number of categories increases, OHE leads to high-dimensionality, namely the dimensionality will be equivalent to the number of unique levels for a categorical variable. High dimensionality increases computational cost when fitting a statistical model. This problem is also referred to as the curse-of-dimensionality. To reduce high-dimensionality caused by applying OHE, dimensionality reduction techniques might be employed, but this might lead to loss of information. Another challenge with OHE is that, when the training set contains a categorical variable for which the unique level set is: $\{b_1,\ldots,b_K\}$, but the test set contains a level not present in this set for the same categorical variable. Since, in the training phase K columns are created encoding categorical variable, a way of encoding the new level should be found. One solution is to assign new levels to the null vector, yet if more than one new level appears in the test set, that is not in the training, more than one level will be assigned to the null vector and collisions occur [5, 17].

As an alternative to OHE, to solve the problem of high dimensionality when the number of unique levels are high, feature hashing is proposed (also called *hash encoding*

or *hashing trick*). This method can reduce the dimensionality, by collapsing multiple levels into one instead of binary encoding each level in a categorical variable. A hash function is used that maps an input to an integer, where different input values can be assigned to the same output and output dimensionality is determined by choice [21]. In the context of categorical encoding, the input value is a level of the categorical variable. After obtaining a reduced representation, one-hot encoding is applied as the final step. Given that a categorical variable has 10 levels, a hash function with output dimensionality of two to represent 10 levels as a two-dimensional vector can be designed. However, the cost of reducing the dimension of a vector might lead to loss of information as two different levels mapped to the same output location are arbitrarily chosen and might not be related in a semantic or syntactic way. Two levels being mapped to the same output is referred to as a collision [22].

A simple method of encoding is integer encoding (also called *label encoding*). This method assigns categorical variables to numeric indices *1* through K, for a categorical variable that contains K levels, where levels are sorted by lexical ordering. A problem with this method is that the numbers assigned to categorical levels are arbitrary and cannot express the significance of a categorical level. Each level in the categorical variable represents a concept that cannot always be meaningfully ordered. Conversely, this method assigns an artificial order to the variable for which any natural ordering may not exist. As an example, if a dataset has a categorical variable consisting of fruit names with values drawn from the set {"*apple*", "*apricot*", "$grapefruit$"}, then, label encoding might assign the labels $\{1, 2, 3\}$ respectively. Adding the representations of apple and grapefruit will not yield another fruit nor comparing the magnitude of their representations provide any information about the size of the fruit.

Another method, ordinal encoding, exists which is sometimes interchangeably used with label encoding, but is slightly different [23]. Ordinal encoding can be used to manually assign integer values to ordinal categorical variables. For instance, a categorical variable that contains the ratings of user experience that has three choices {"$excellent$", "*okay*", "$awful$"}, with ordinal encoding, can be encoded as $\{3, 2, 1\}$. A

potential problem with this method is that, “okay” is twice better than “awful” and “excellent” is 1.5 times better than “okay”. The magnitude of relationship between categories may not be proportionate. Another problem arises, as the number of categories increases, this method becomes infeasible, since the amount of manual work increases as well [24].

Frequency encoding is a method that maps the frequency distribution of categories as the new value for each category. Precisely, for a level of a categorical variable frequency distribution refers to the ratio of the count of the level (the number of times the level occurred in the categorical column) and the number of instances in the dataset. In case there is a functional relationship between the frequency of the categories and the target variable, this method can be useful. It can reduce the total number of levels by mapping different categories to the same label if their frequency is the same. In a way it categorizes categories as highly frequent, frequent, rare and so on. When there are categories that do not appear in the training set, but are observed in the test set, new categories might be encoded with a frequency of one or mapping the least observed frequency during training might be another option [18].

Another approach to reduce the cardinality is by performing clustering. Clustering reduces the number of categories, where total number of unique categories are K, to L categories where $L << K$. After reduction to a set of L unique values, OHE can be performed. An approach for clustering is grouping values that exhibit similar target statistics. Hierarchical clustering uses the difference of the statistics of the target as probability or average as distance metric between groups [19]. Quinlan (1985) proposed to use information-theoretical metric like gain ratio to measure the effect of merging each possible pair of clusters. Two clusters with the greatest improvement of the gain ratio are merged. This process is iterated until no improvement is achieved or the set numbers of clusters are reached [25]. Carrizosa, Restrepo and Morales (2021) propose a method to cluster the levels of the categorical variables by an iterative process where levels of a categorical variable is first ordered and feasible clusterings are found where the number of clusters is set to two. Then, a Greedy Randomized Adaptive Search

Procedure (GRASP) is used, which randomly selects between the feasible clusterings with the highest out-of-sample accuracies. This method is used to determine which cluster is the best for the given categorical variable [13].

Using target statistics (target encoding) to encode categorical variables is also a popular method. Proposed by Micci-Barreca [26], each level is encoded given the effect it has on the target variable. The basic principle for this encoding scheme is to map each categorical value to an estimate of the probability or the expected value of the target. The new variable generated by using target statistics, is considered as a new numerical feature instead of using OHE later on. Therefore, the new variable created is one-dimensional, the number of categorical variables in the original dataset remains the same. Rare categories are taken into consideration and a smoothing approach is adopted, where the estimate of the target given the category gets blended with the prior probability of the target. For a category with higher frequency, the estimate of the target is weighted towards the value of the target given the category. For rare categories, estimate is weighted toward the overall baseline for the target. CatBoost uses target statistics to represent categorical variables, providing an alternative approach to smoothing and overfitting avoidance. Relying on the ordering principle, inspired by online learning algorithms that get their training algorithms sequentially in time. Hence, the values for the target statistic rely only on the observed history [27].

Pargent, Pfisterer, Thomas and Bischl (2022) [8] has published a benchmarking study on encoding high-cardinality categorical variables comparing various methods. They proposed GLMM encoding, which is a generalization of smoothed target encoding where a linear mixed model in which the target is predicted by a random intercept for each feature level in addition to a fixed global intercept is used. GLMM encoding is comparable to smoothed target encoding, however, is slower in terms of run time. They also used tree based leaf encoding algorithm where, a decision tree is fitted on the categorical variable of interest and the target variable and each level is encoded by the number of the terminal node that it resides in. Leaf Encoding categorizes levels with similar target values. Later newly encoded variable is one-hot encoded as the terminal

nodes are considered to be nominal. New levels and missing values are encoded with the number of the terminal node with most observations during training. They reported that Target Encoding and GLMM Encoding yields the best results compared to other encoding strategies used in the study [8]. Another tree based method is random forest encoding, for which a random forest is trained to predict the target based on the single categorical feature. For each observation in the training set, only the trees for which the observation was out-of-bag is used to compute the prediction. The whole forest is used for the encoding during prediction phase. For new levels, the average prediction across all leafs for each tree before averaging across all trees are predicted [20].

Categorical split encoding uses the path of a level as the vector representation obtained from classification and regression trees (CART). It is similar to cluster encoding as it assigns similar representations to levels with similar paths when traversing a decision tree. If a level ended up in the same terminal node as another level their representations will be the same. Another advantage is that, compared to one hot encoding the number of levels are reduced and as a result computational complexity is reduced. Missing values are also imputed by making use of surrogate split mechanism, without the need to impute after encoding. Most categorical variable encoding strategies cannot handle missing values innately and as the percentage of the missing values increases, imputing becomes an important issue.

3. THEORETICAL BACKGROUND

In this chapter, theoretical background that is needed to understand the following chapters is presented. Firstly, an in-depth understanding of classification and regression trees (CART) is provided [14], then, the concept of surrogate splits is explained extensively.

3.1. CART

Breiman, Friedman, Olshen and Stone (1984) [14], have introduced the term classification and regression trees (CART), which stands for algorithms that can be used for classification and regression problems. Regression problems are where the target variable $y \in \mathbb{R}$ and classification problems are where the target variable y is from a finite set $\{1, ..., J\}$. When $J = 2$, the task is binary classification, when $J > 2$, the task is multiclass classification. CART operates by recursively partitioning the feature space into two descendant subsets at each iteration and is represented graphically as a binary decision tree. Each partitioned region is assigned a class label [28]. In Figure 3.1 a decision tree is given, for a binary dataset that has two independent variables, X_1 and X_2, where $0 \leq X_i \leq 10$ and $i = 1, 2$.

In the following paragraphs, the terminology associated with decision trees is walked through and how a decision tree partitions feature space into rectangles or subsets is demonstrated in a precise way. In Figure 3.1, each individual box refers to a node and each node is denoted with t_n where $n = 1, 2, 3, 4, 5$, since there are five nodes. Nodes can be broadly classified as terminal and non-terminal nodes. Terminal nodes are designated by a class label and non-terminal nodes continue splitting process. Any node that is stemming from another node is called the child node of that node, and the node that stems into new nodes are called the parent node of them. For classification trees, the majority class in the leaf nodes specify the target value of the observations that fall into that node and for regression trees the mean of the leaf node

specify the the target value for the observations that fall into that node. Multiple leaf nodes with the same class label can exist. The predicted labels are obtained by putting together all leaf nodes with the same class. In particular, nodes can be classified into three categories: root nodes, decision nodes and leaf nodes. In Figure 3.1 the root node is labeled with t_1. Root node is located at the top of the decision tree and all the other nodes originate from the root node. Node t_3 is a decision node, these are intermediate nodes which either branch out to another decision node or branch out to a leaf/terminal node. Nodes t_2, t_4 and t_5 are leaf nodes, which are also referred to as the terminal nodes. Terminal nodes are indicated by a rectangular box and non-terminal nodes are indicated by a circle in Figure 3.1. The class labels are from the set $\{1, 2\}$, hence, a binary classification problem. Right below the terminal nodes, t_2, t_4 and t_5, corresponding class labels are shown. Leaf nodes t_2 and t_4 correspond to label *2* and leaf node t_5 corresponds to label *1*. Two splits are made to construct the tree, the first split is made on the variable X_1, where majority of the class labels are *2* if $X_1 \leq 6$ and otherwise majority of the class labels are separated as *1*. The second split is made on the variable X_2 where when $X_2 \leq 7$, the majority of the instances with class label *2* are sent to the left and otherwise to the right.

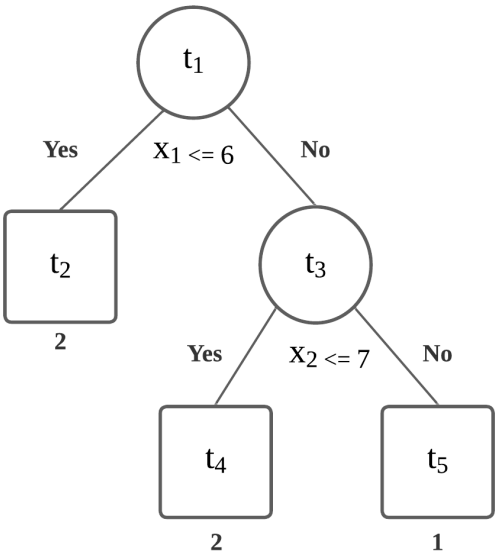

Figure 3.1. Binary decision tree visualization.

Another way to represent the binary tree in Figure 3.1 is shown in Figure 3.2. As given, the tree is constructed on two numerical features X_1 and X_2. Feature X_1 is on x axis and feature X_2 is on the y-axis in Figure 3.2. There are 19 instances or observations in the dataset, each shown with a star or a circle, depending on the label of the instance. The tree procedure that recursively partitions the feature space generated by X_1 and X_2 can be clearly observed. Feature space is divided into three rectangular regions, t_2, t_4 and t_5. The splitting points are marked in the axes for each variable. These regions correspond to the terminal nodes of the binary tree in Figure 3.1. The regions t_2 and t_4 majorly consist of the instances with label *2* and region t_5 consists solely of instances with label *1*. As one can observe, the three regions in Figure 3.2, make the feature space more homogeneous in terms of class distribution. Region t_5 consists solely of the class label *1* and hence, is called homogeneous or pure. It will be discussed why the aim is to have pure terminal nodes, or better put, a purer node is aimed when choosing the splits. The area of two terminal nodes combined give the area of the parent node, for instance the combined region for t_4 and t_5 give the area for the node t_3, which is the parent node of t_4 and t_5, for the tree in Figure 3.1 [14, 28, 29].

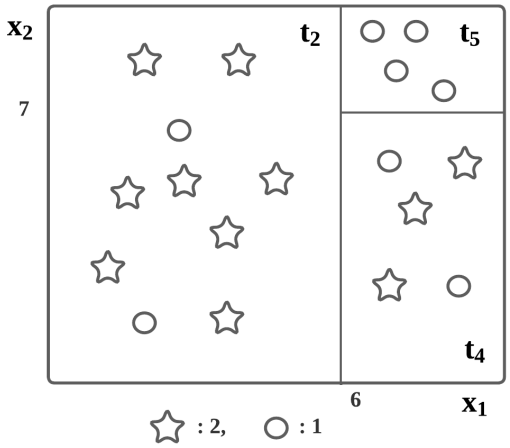

Figure 3.2. Geometric viewpoint of the binary tree.

Morphological elements and recursive partitioning of the feature space is demonstrated in the former paragraphs. However, some key elements are missing and should be cleared up, as to why the variable X_1 is the initial split and therefore at the top of the tree. Why is the variable X_1 split on the threshold $X_1 \leq 6$? Next step is to

provide an explanation on how a decision tree is constructed. This subject brings up the questions:

(i) How to decide which variable should determine the split at each iteration/split?
(ii) How to decide when to terminate the tree and call it a leaf node or continue splitting the tree?
(iii) How to decide which label should be assigned at the terminal nodes?

3.1.1. Splitting Mechanism

For each problem, a feature space $\boldsymbol{X}$ exists, which has M features, $X_1, ..., X_M$. Feature space is a mixture of continuous and categorical types. The rest of the paragraph is a mere summary of the splitting procedure. For tree growing procedure, initially for each variable in the feature space, a set of binary questions are asked. For instance, in Figure 3.1 for the first split, the question "Is X_1 less than or equal to 6?" is asked and based on the answer, the instances are sent to left child node or right child node. To find the optimal question that gets us to the best split, a goodness of fit measure is set. The goodness of fit measure depends on the impurity of the potential split. If the impurity of the split is the lowest, then that question is selected for the current variable. Then, out of all the variables the one with the lowest impurity and maximum goodness of split is chosen as the first variable needed to split on and it is located at the root node of the decision tree.

Let us elaborate, on the binary set of questions. Denoting the set of all questions that a variable can take with $\boldsymbol{Q}$. If the independent variable x_m is continuous, then the set of all binary question for x_m is in the form:

$$\mathbf{Q} = \{Is\ x_m \leq c?\}\ , c \in (-\infty, \infty). \tag{3.1}$$

If the independent variable x_m is categorical, where the unique values of x_m form the set $\{b_1, ..., b_K\}$,then the set of all binary question for x_m is:

$$\mathbf{Q} = \{Is\ x_m \in A?\} \tag{3.2}$$

where A is any subset of the set $\{b_1, ..., b_K\}$. The set of questions for each variable are also referred to as the potential splits, that is needed to choose one from. The number of splits for a continuous variable, is determined as follows. Continuous variable x_m is ordered in an ascending manner, then midpoints of two adjacent values are evaluated to determine the set of questions. Maximum number of splits can be at most N, where N is the number of instances in the dataset. So that set of the binary questions is:

$$\mathbf{Q} = \{Is\ x_m \leq c_n?\}\ , n = 1, .., N^{'} \leq N. \tag{3.3}$$

In Equation 3.3, c_n is the halfway between each adjacent value, when the continuous variable is sorted in ascending order. For categorical variables, there are $2^{K-1} - 1$ possible splits for a K level categorical variable, since the direction of the split is reversible. When the set of all possible questions/splits for all the variables in the dataset is computed, for each node all the features are checked for the best split. Initially, the best split for each individual feature is found, then for each variable the best split is compared and the best variable to make the split on is selected. How the best split is selected is explained in the following paragraphs.

To choose the best split for inter and intra variable level, the goodness of split criterion is used. The goodness of the split criterion decides on the best split by choosing minimum impurity. Most possible purity in the terminal nodes is desired, because higher purity indicates better possibility of the final decision being correct. As a result, obtaining purer nodes is the aim at every iteration of the tree. The impurity function is a function of probabilities of the splits made belonging to the classes $1, ..., J$ for a J class classification problem. The percentages of the split groups are used as the probabilities. Impurity function at any node t can be denoted with:

$$i(t) = \phi(p(1|t), ..., p(J|t)). \tag{3.4}$$

The impurity function can be any function that has the following properties:

- achieves its maximum when the probability of all the classes are equal
- has the minimum value when the probability of a class is 1 and other class probabilities are 0
- is a symmetric function

The goodness of the split or the decrease in impurity can than be notated as:

$$\mathbf{\Delta I}(s,t) = [i(t) - (p_R\, i(t_R) + p_L\, i(t_L))]\, p(t) \tag{3.5}$$

where s is the split of node t, p_L is the percentage of the instances that are sent to t_L and p_R is the percentage of the instances that are sent to the t_R and $p(t)$ is the proportion of the dataset that correspond to node t. Basically, the decrease in impurity between parent and child node is calculated. The best split is determined by finding the variable that yields largest decrease in impurity, that maximizes $\mathbf{\Delta I}(s,t)$. The two most widely used impurity functions are entropy and gini index.

3.1.2. Knowing When to Terminate

Tree splitting procedure can be continued until there is one terminal node for each instance in the dataset. However, a stopping criterion is needed to prevent overfitting. The stopping criterion should not also create too shallow of a tree as this will lead to underfitting. One strategy is to stop splitting, when the maximum amount of increase in impurity is less than the priorly set threshold β:

$$\max\ \mathbf{\Delta(i)}(s,t) < \beta. \tag{3.6}$$

This strategy is not the most efficient one, as one step ahead at a time when growing the tree can be seen. Current split may result in a little decrease in impurity but the next split might be considered a good split. If the threshold value is set too low, the tree might grow too deep or if the threshold value is set too high, the growth of the tree might be terminated early. This is a pre-pruning method as the tree is being pruned

simultaneously while growing the tree. Breiman, Friedman, Olshen and Stone (1984) showed that it is better to prune the tree after letting the tree grow without interfering. After the tree is at its full size, pruning upward or selecting the best subtree in the fully grown tree is suggested. One strategy to prune is using the cross-validation error or test samples to choose the subtree having the least estimated true misclassification rate. With large sample sizes use of an independent test sample is suggested and with small sample sizes checking the cross-validation error is suggested [14].

3.1.3. Class Assignment of the Leaf Nodes

For classification problems, where the target variable is from the set $c \in \{1, ..., J\}$. Terminal nodes are denoted with $T^{'}$. Our aim is to assign a class label to each terminal node, where $t \in T^{'}$ and the assigned class can be shown with $c(t)$. For each terminal node in the tree, the majority class should be assigned. This can be denoted with:

$$c(t) = arg \max_{c} \; p(c|t). \tag{3.7}$$

If the majority is a tie between two or more classes, the class label is assigned to one of the majority tie classes at random. For instance, in Figure 3.2, for terminal node t_2 the count of label *2* is eight and the count of label *1* is two. Label *2* is the majority label in t_2, hence, all the instances in that region are assigned the label *2*. For regression trees the class assignment is determined by taking the mean of the labels of the instances in the terminal node.

3.2. Dealing With Missing Values in Decision Trees: Surrogate Splits

The problem of missing values can be solved through the use of surrogate splits in decision trees without the need to do pre imputation on the dataset. Using surrogate splits for the missing instances can account for mainly two cases: if the training set has missing values and if an unseen instance is given to the model that has missing values. For a node t of a decision tree, the surrogate split of the best primary split gives the most similar split to best primary split. To compute the most similar variables to

the best split variable a measure of similarity is defined. In case the variable with the best split has a missing value, the best surrogate split for it can be used to decide whether the value goes to t_L or to t_R. Similarly, second best or third best surrogate variables can be defined for a best split primary variable and can be used in case the first best surrogate split has missing a value as well. The best surrogate split is the one that predicts the action of the best split most accurately, that is done by finding the variable that splits the data points most similar to the best split. The following paragraph describes how the surrogate splits are selected [14].

Let Δ^* be the best split at any node t that splits t into t_L and t_R. For any variable x_m, other than the best split variable, the set of all splits of x_m can be denoted with S_m and the set of all complementary splits can be denoted with $\overline{S_m}$ that branches $t^{'}$ into $t_L^{'}$ and $t_L^{'}$. If the split is $\{Is \;\; x \in A?\}$, then the complementary split is $\{Is \;\; x \in A^c?\}$. Any split on variable x_m can be denoted with Δ_m, which is formed by $S_m \cup \overline{S_m}$. The aim is to find the split on x_m that sends the most number of instances in common with the best split to left node and right node. The probability that an instance falls into $t_L \cup t_L^{'}$ is denoted with:

$$p(t_L \cup t_L^{'}) = \sum_j \pi\left(j\right) N_j\left(LL\right) / N_j \tag{3.8}$$

where $\pi(j)$ is the prior probability of class j, that is formulated as N_j/N. $N_j(LL)$ is the number of cases that both Δ^* and Δ_m send to t_L. N_j is the number of samples in class j. Then, the estimated probability that both Δ^* and Δ_m send an instance to the left can be expressed with:

$$p_{LL}(\Delta^*, \Delta_m) = p(t_L \cup t_L^{'})/p(t) \tag{3.9}$$

where $p(t)$ is the probability of any sample going to node t regardless of its class. The probability of any sample going to node t can be formulated as:

$$p(t) = \sum_j \pi(j) N_j(t)/N_j \tag{3.10}$$

where $N_j(t)$ is the number of samples of class j going to node t.

Then, the formula for the estimated probability that both Δ^* and Δ_m send an instance to the left can be simplified as:

$$p_{LL}(\Delta^*, \Delta_m) = \frac{\sum_j N_j(LL)}{N(t)}. \tag{3.11}$$

The estimated probability is simplified to the ratio of the number of instances that both Δ^* and Δ_m send left and total number of instances in node t. Similarly for t_R, the same definitions apply. As described, the surrogate variable can be used to predict the behaviour of the best split variable. For instance, for any node, if the best surrogate variable sends an instance to $t_L^{'}$, then it is predicted that the best split is going to the send the instance to t_L. This is handy when the value of the best split variable is missing, hence, the surrogate splits behaviour is used to predict the best split. The probability that Δ_m predicts Δ^* correctly is formulated as:

$$p(\Delta^*, \Delta_m) = p_{LL}(\Delta^*, \Delta_m) + p_{RR}(\Delta^*, \Delta_m). \tag{3.12}$$

Out of all splits on x_m, the aim is to find the split that maximizes the probability that Δ_m predicts Δ^* correctly. Then that split will be the best surrogate split for variable x_m. The best surrogate variable for x_m is denoted with $\tilde{\Delta_m}$. The formulation of the selection of best surrogate split for variable x_m is:

$$p(\Delta^*, \tilde{\Delta_m}) = \max_{\Delta_m} p(\Delta^*, \Delta_m). \tag{3.13}$$

The method to choose the best split for a variable x_m is described above. To predict, which variable is the best predictor for Δ^*, predictive measure of association between Δ^* and $\tilde{\Delta_m}$ is checked. Predictive measure of association is formulated as:

$$\lambda(\Delta^*|\tilde{\Delta_m}) = \frac{min(p_L, p_R) - (1 - p(\Delta^*, \tilde{\Delta_m}))}{min(p_L, p_R)}. \tag{3.14}$$

The best surrogate split yields the maximum predictive measure of association. Instead of using surrogate split, one could use the relative probability that an instance will be sent to t_L or t_R. The probability that the instance will be sent to t_R is denoted with p_R and the probability that an instance will be sent to t_L is denoted with p_L. Then,

when a new instance is to be predicted, one can use the relative majority probability $\max(p_L, p_R)$, if the instance has a missing value in the best split variable. That is, if the $p_L = \max(p_L, p_R)$, then the instance is sent to t_L. The error probability of sending the instance to the wrong node is $\min(p_L, p_R)$. When using surrogate splits the aim for the error is to be much less than $\min(p_L, p_R)$. Hence, predictive measure of association is formulated in a way that, relative reduction in error is yielded when the surrogate split is used predict the best split compared to the error $\min(p_L, p_R)$. The best surrogate split is chosen with $\max \lambda(\Delta^* | \tilde{\Delta}_m)$. If the relative reduction of error is less than zero, the variable is discarded as surrogate variable. The second best surrogate variable can be chosen as the variable that gives the second best predictive measure of association, this applies for the third, fourth surrogate variable and so on.

4. METHODOLOGY

In this chapter, proposed approach to encode categorical variables using categorical split encoding is presented. Using a tree based approach, categorical split encoding uses primary and surrogate split information of a categorical variable, in case this information exists. How categorical split encoding works is demonstrated in the following paragraphs.

The categorical variable presented in the tree, in Figure 4.1, can be either one of the best split, a competing split or a surrogate split variable at node t. It consists of five unique levels: $\{a, b, c, d, e\}$. All the levels are available at the root node. After the first split, levels a and b are sent to t_L and levels c, d and e are sent to t_R. At the terminal nodes, levels a, b and e are perfectly separated, while c and d reside in the same terminal node. The class labels corresponding to terminal nodes are shown underneath the terminal nodes. If the path each level takes is tracked, the representation in Table 4.1 is obtained.

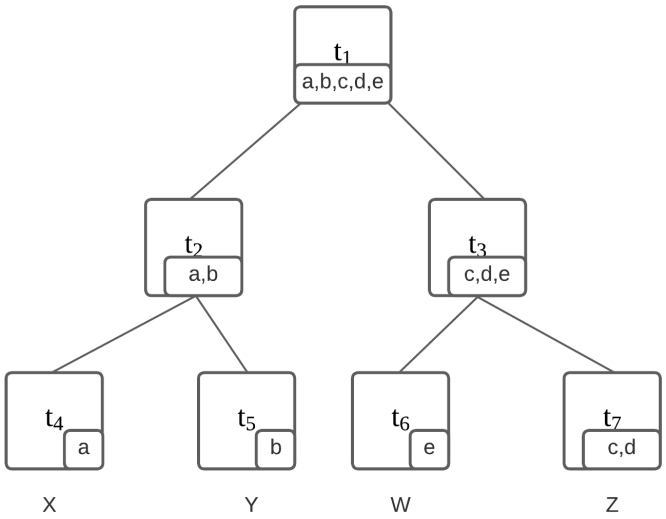

Figure 4.1. Path of a categorical variable.

In Table 4.1, columns one to three represent the node numbers that are not the terminal nodes. The row values are the levels of the categorical variable. For

each non-terminal node which direction the level is sent to is marked in the table. L denotes that the level is sent to the left child node and R denotes that the level is sent to the right child node. $-$ denotes the levels that are not present in that node. In Table 4.1, missing levels at nodes 2 and 3 are denoted with $-$ can be observed. By checking the table laterally, it can be seen that level a is represented by $LL-$ and level b is represented with $LR-$. Level b is more similar to a than the remaining levels as their representations are similar. The representations for levels c and d are the same, hence, they end up in the same terminal node. Categorical split encoding, for this instance, initially generates three columns based on the 3 non-terminal nodes, mapping the corresponding direction for the level of concern. After generating three columns, columns that contain missing values or contain instances denoted with $-$, are dummy encoded, taking $-$ as the reference level. The resulting representation after performing categorical split encoding is shown in Table 4.2.

Table 4.1. Representations of each level.

	t_1	t_2	t_3
a	L	L	-
b	L	R	-
c	R	-	R
d	R	-	R
e	R	-	L

In Table 4.2, as the first node does not contain any missing levels (levels denoted with -), it is not dummy encoded (represented with t_1). Levels denoted with L are mapped to 0 and levels denoted with R are mapped to 1, hence, they are integer encoded. Representations in the second node and the third node contain missing levels as the level is not present, because it was sent to a different node. It can be observed in Table 4.2 that the columns for node two and three are dummy encoded (represented with $t_2.L$, $t_2.R$, $t_3.L$, $t_3.R$ respectively). Hence, five columns are generated as the output of categorical split encoding.

Each node in the tree can create more than one column for a categorical variable, in case the categorical variable is present as a best, competing or a surrogate split in

that node. Each node can create at most two columns. The final representation is then used as the representation for the categorical variables prior to fitting a statistical model or analysis of the dataset after original categorical variables are dropped.

Table 4.2. Categorical split encoding representation.

	t_1	$t_2.L$	$t_2.R$	$t_3.L$	$t_3.R$
a	0	1	0	0	0
b	0	0	1	0	0
c	1	0	0	0	1
d	1	0	0	0	1
e	1	0	0	1	0

Categorical split encoding also encodes missing values from surrogate splits. As mentioned in Breiman, Friedman, Olshen and Stone (1984), surrogate splits imitate the behaviour of the best primary split variable when the best split variable is missing. Similarly when encoding with categorical split encoding, if the categorical variable is the best split variable, missing values are imputed from surrogate variables starting with the one with the highest agreement level. If the surrogate variable is numerical, then all the missing values can be imputed. If the categorical variable is a competing variable and has missing values, the columns with the missing values cannot be encoded and are dummy encoded. If the categorical variable is a surrogate variable, it can be encoded from the best split variable. This method reduces the greediness of the decision tree, while clustering similar values together based on the path they take, overfitting aspect of the trees are mitigated by encoding missing values.

Leaf encoding results in a different representation. It uses the number of the terminal node that a level resides in as its representation. Since the number of a terminal node is not meaningful, the resulting representation is one-hot encoded. In Table 4.3 resulting representation of leaf encoding is shown. As observed, four columns are generated. The main difference between categorical split encoding and leaf encoding is that leaf encoding creates an encoding by fitting a decision tree to one categorical column and the target variable at a time (the number of decision trees created is equal to the number of the number of categorical columns) and categorical split encoding

creates encoding by fitting a decision tree to the whole dataset (one decision tree is created). Leaf encoding does not utilize numerical columns while categorical split encoding does. Leaf encoding only takes into the consideration where a categorical ends up and categorical split encoding uses the path of a categorical variable.

Table 4.3. Leaf encoding representation.

	t_4	t_5	t_6	t_7
a	1	0	0	0
b	0	1	0	0
c	0	0	0	1
d	0	0	0	1
e	0	0	1	0

In Figure 4.2, pseudocode for categorical split encoding is provided. A brief explanation of the pseudocode is given in the following paragraphs. As the first step, a decision tree is fit on the training set (line 1). As the impurity function gini impurity is used, maxdepth is set to four, complexity parameter is set to zero and use of surrogate splits is enabled with the option to send the observation in majority direction if all surrogates are missing. The other parameters are included as default. Increasing maxdepth parameter can capture more intricate patterns about the data, however increasing it past a certain depth might also lead to overfitting, where useful patterns are not captured but training set is memorized instead. Since complexity parameter is set to zero, a split that improves the relative error positively, is not pruned off. Use of surrogate splits, are used to split the instances missing the best split variable. Training set consists of a mixture of numerical and categorical variables. Each categorical variable is from a finite set of levels, $\{b_1, ..., b_K\}$ (lines 1-3). For each node of CART two main variables are defined:

(i) ordered set of primary split variables
(ii) ordered set of surrogate split variables

where ordered set of primary splits and ordered set of surrogate splits are denoted with S and S' respectively. Ordered, in this context means that the sets are ordered in

descending order of their goodness of fit for the primary split set and predictive measure of association for surrogate set (lines 4-8). For each categorical variable in each node, the primary split and surrogate split information is utilized, the path that the levels in the categorical variable take are mapped to the levels in categorical variable (lines 9-17). After the initial path based encoding frame is obtained, it is processed. If any column in the encoding frame contains null values, imputation is carried out. Imputation can only be carried out for the best primary split variable and best surrogate split variable. The best primary split variable is imputed from the best surrogate variable. The best surrogate split variable is imputed from the best primary split variable. After imputation, if any column in the encoding frame contains $-$, that column is dummy encoded where $-$ is the reference level (lines 18-23). The output of CSE is the encoded categorical dataset denoted with $D_{cat}^{train'}$. This categorical dataset is then appended to the original dataset where the original categorical columns are dropped.

Input Fit CART on D^{train}
Output $D_{cat}^{train'}$

1: $D^{train} = (x_1, ..., x_M)$
2: $D_{num}^{train} = \forall\, x_m \in D^{train} \ni x_m \in \mathbb{R}$
3: $D_{cat}^{train} = \forall\, x_m \in D^{train} \ni x_m \in \{b_1, ..., b_K\}$
4: **procedure** ITERATE($CART$)
5: **for each** $t \in T$ **do**
6: $S = \{x_1, ..., x_r\} : set\ of\ primary\ split\ variables$
7: $S^{'} = \{1, ..., x_b\}\ where\ S^{'} \in D^{train} \ni x_y \in S^{'}$:
8: $\lambda(\Delta^*, \Delta_y^{'}) = \frac{min(p_L, p_R) - (1 - p(\Delta^*, \Delta_y^{'}))}{min(p_L, p_R)}\ if\ \lambda(\Delta^*, \Delta_y^{'}) > 0$
9: **for each** $x_c \in\ t\ where\ (x_c \in D_{cat}^{train})\ \&\ (x_c \in S \cup S')$ **do**
10: **for each** $b_d \in x_c$ **do**
11: **if** $b_d \in t_L$ **then**
12: $D_{cat}^{train}[x_c].replace(b_d, \mathrm{L})$
13: **else if** $b_d \in t_R$ **then**
14: $D_{cat}^{train}[x_c].replace(b_d, \mathrm{R})$
15: **else if** $b_d \notin \{t_L, t_R\}$ **then**
16: $D_{cat}^{train}[x_c].replace(b_d, \text{-})$
17: **return** $D_{cat}^{train'}$
18: **procedure** PROCESSING($D_{cat}^{train'}$)
19: **for each** $x_c \in D_{cat}^{train'}$ **do**
20: **if** $(NA \in x_c)\ \&\ (x_c = S_1 | x_c = S_1^{'})$ **then**
21: $x_c^{'} = Impute(x_c)$
22: **if** $- \in x_c$ **then**
23: $x_c^{'} = Dummy(x_c)$

Figure 4.2. Pseudocode of categorical split encoding.

4.1. Illustrative Example

To demonstrate how categorical split encoding works, a toy dataset is generated. The first 10 instances of the dataset is given in Table 4.4 and it consists of 50 instances. The feature space has four numerical features, a categorical feature and a binary target column. The categorical column is created by assigning leaf indices as the levels using an ensemble of totally random trees. Columns one through four are numerical variables, column $X5$ is the categorical variable and y is the target variable. Categorical variable $X5$ consists of 28 unique levels and is from the set of alphabetical letters. Dataset does not contain any missing values.

Table 4.4. Toy dataset.

X1	X2	X3	X4	X5	y
-1.06441	-0.25032	-0.70212	-0.81798	a	1
-0.12938	-0.04868	0.048378	-0.6861	a	0
-0.34454	2.734048	2.528541	-2.18223	b	0
1.113819	-0.04219	-0.29796	1.594023	c	1
-1.52744	-1.93968	-1.09145	-0.09653	a	0
0.916548	1.796848	0.114019	-0.15196	d	1
2.012829	-1.57125	-0.54618	1.060407	e	1
-2.27319	0.529323	1.860576	-1.55657	f	0
-2.62657	-0.79046	0.504952	-1.95929	g	1
-1.735	-1.49649	-0.93098	-1.47797	g	1

A decision tree is fit to the dataset and the summary of the tree is obtained. Three splitting nodes: t_1, t_2 and t_5 are yielded by the tree. The output of categorical split encoding is given in Table 4.5. The representation of categorical column $X5$ yields seven columns after categorical split encoding.

Columns names consist of four parts (or three, depending on whether any missing levels exist in the current node), each separated by a dot. For instance, the first column is $t_1.X5.P$, here, t_1 is the splitting node number, $X5$ is the name of the categorical column, P indicates whether the categorical variable is a primary or surrogate split in the current node. The second column $t_2.X5.P.L$ has an extra component at the end, L, which indicates that at the current node, some levels are missing, hence, the column

is dummy encoded. L represents the levels that are sent to the left child node. Also, the first column, $t_1.X5.P$, is mapped to integers before fitting a statistical model.

Table 4.5. Categorical split encoding representation.

$t_1.X5.P$	$t_2.X5.P.L$	$t_2.X5.P.R$	$t_5.X5.P.L$	$t_5.X5.P.R$	$t_5.X5.S.L$	$t_5.X5.S.R$
L	0	1	0	1	1	0
L	0	1	0	1	1	0
L	1	0	0	0	0	0
R	0	0	0	0	0	0
L	0	1	0	1	1	0
R	0	0	0	0	0	0
R	0	0	0	0	0	0
L	0	1	0	1	0	1
L	0	1	1	0	1	0
L	0	1	1	0	1	0

To demonstrate how the representation in Table 4.5 is obtained, the sample output from CART summary is given, for t_1 in Figure 4.3 [30]. It is observed that $X5$ is the best split variable. $X1$, $X4$, $X3$, $X2$ are competing variables. Surrogate variables of $X5$ are $X1$, $X4$, $X2$, $X3$, respectively. Since the split on $X5$ does not have any levels that are missing yet, a column with L and R values is created for t_1. The directions that are lined up next to $X5$ is in the alphabetical order of the levels. For instance, the level "a" corresponds to L. This node is represented by column $t_1.X5.P$ in Table 4.5.

```
Node number 1: 50 observations,    complexity param=0.68
  predicted class=-1  expected loss=0.5  P(node) =1
    class counts:    25    25
   probabilities: 0.500 0.500
  left son=2 (33 obs) right son=3 (17 obs)
  Primary splits:
      x5 splits as  LRLRRRRLLRRLLRLLLRRLLRRLRRLL, improve=12.878790, (0 missing)
      x1 < -0.06693635 to the left,  improve= 8.047619, (0 missing)
      x4 < -0.9492213  to the left,  improve= 4.166667, (0 missing)
      x3 < 0.08901588  to the left,  improve= 2.011494, (0 missing)
      x2 < -0.4541895  to the left,  improve= 1.972625, (0 missing)
  Surrogate splits:
      x1 < -0.06693635 to the left,  agree=0.88, adj=0.647, (0 split)
      x4 < 0.1914634   to the left,  agree=0.82, adj=0.471, (0 split)
      x2 < 1.171144    to the left,  agree=0.74, adj=0.235, (0 split)
      x3 < -3.08019    to the right, agree=0.70, adj=0.118, (0 split)
```

Figure 4.3. Sample output of CART summary (t_1).

To explain columns four through seven in Table 4.5, CART output for node t_5 is given in Figure 4.4 [30]. In t_5, $X2$ is the best split variable. Categorical variable $X5$ is the competing split with the least improvement and the surrogate split with the least predictive measure of association. There are missing levels that are denoted with -. Since $X5$ exists as a primary and a surrogate competitor, initially two columns need to be generated. Since, missing levels exist in both surrogate and primary splits, the columns are dummy encoded (- is set as reference level and dropped). Hence, we obtain four columns as the representation of $X5$ in t_5.

```
Node number 5: 22 observations,    complexity param=0.06
  predicted class=-1  expected loss=0.3636364  P(node) =0.44
    class counts:    14     8
   probabilities: 0.636 0.364
  left son=10 (15 obs) right son=11 (7 obs)
  Primary splits:
      x2 < -0.4404852  to the left,  improve=2.52467500, (0 missing)
      x3 < 0.08901588  to the left,  improve=2.52467500, (0 missing)
      x1 < -1.117373   to the left,  improve=0.19891220, (0 missing)
      x4 < -0.9492213  to the left,  improve=0.19891220, (0 missing)
      x5 splits as  R------RL--R---------------, improve=0.02797203, (0 missing)
  Surrogate splits:
      x1 < -0.2220568  to the left,  agree=0.773, adj=0.286, (0 split)
      x3 < 0.1797609   to the left,  agree=0.773, adj=0.286, (0 split)
      x5 splits as  L------RL--L---------------, agree=0.773, adj=0.286, (0 split)
```

Figure 4.4. Sample output of CART summary (t_5).

5. EXPERIMENTS

Experiments are coded in open source statistical software R [30]. All materials including reproducible code can be accessed through the github repository [31]. Experimental details are outlined in this chapter. First, the implementation of categorical variable encoding strategies used are discussed in detail. Second, the summary of the datasets used in experiments are provided. Finally, details about hyperparameter tuning, machine learning algorithms and evaluation metrics are ellucidated.

5.1. Encoding Strategies

Encoding strategies used are: integer encoding, one-hot encoding, frequency encoding, leaf encoding, target encoding and categorical split encoding. How the methods are implemented are explained in the following subsections.

5.1.1. Integer Encoding

For an K level categorical column in the training set, the levels are numbered from *1* to K after sorting the levels alphabetically. The numbered levels are then mapped to the levels in the test set. If test set contains a new level that does not exist in the training set it is assigned as a missing value. Missing values are imputed with the mean of the categorical column after encoding.

5.1.2. OHE

To one-hot encode categorical variables, caret R package is used [32]. Training sets and test sets are separately one-hot encoded. Then, the mutual columns between the resulting frames are filtered, if any column is only present in either training or test set, it is dropped. This solves new level in the test set problem, since only mutual columns are used. Missing values are imputed from the column mean after encoding.

For some high-cardinality datasets, OHE was not performed, as for those datasets performing OHE is computationally demanding and takes too much time.

5.1.3. Frequency Encoding

The ratio of the frequency of each level and the number of instances in the training set is mapped as the representation of the level. New levels in the test set are assigned to be missing. Missing values are imputed with the mean of the column after encoding.

5.1.4. Leaf Encoding

Leaf encoding as proposed by Pargent, Pfisterer, Thomas and Bischl (2022) is used [8]. The code uses rpart package in R, to build decision trees. For each categorical variable a decision tree is build, where categorical feature is the predictor and the target variable is the independent variable. The trees are pruned using cross-validation error, then the levels are assigned to the number of the terminal node in the tree. Terminal node numbers are nominal and the resulting frame is one-hot encoded as the result. Missing values and new values are both encoded with the number of the terminal with the most number of observations.

5.1.5. Target Encoding

Smoothed target encoding as proposed by Micci-Barreca [26] is used. This method calculates the probability estimates as the mixture of the posterior probability of the target variable given a categorical level and the prior probability of the target variable based on the training set. These two probabilities are blended using a monotonically increasing function λ, which is a function of the sample size. New values are assigned as a missing value. Missing values are imputed with feature mean after encoding.

5.1.6. Categorical Split Encoding

For categorical split encoding, an encoding matrix is obtained by fitting a decision tree to the dataset. For this, rpart package in R is used. Complexity parameter is set to zero and maxdepth is set to be four for all datasets when using rpart. The use of surrogate splits is enabled. Summary of the tree is parsed for each node that contains primary and surrogate split information. Categorical variables are represented with the direction of their individual levels. Missing values are imputed for the columns that are either the best primary split or the best surrogate split. New values in the test set are assigned as missing. If any missing values exist in the encoding frame, imputation is performed from the mean of the column with missing values. After the encoding process, if a variable with $-$ exists in a created column, dummy encoding is applied to the column.

5.2. Datasets

A total of 15 datasets are used, 11 from classification setting (eight binary classification and three multiclass classification) and four from regression setting, which are available in OpenML platform [33]. Description of the datasets is given in Table 5.1. OpenML ID is given in the first column, with this ID, datasets can be downloaded for further use from OpenML platform. Name of the dataset is given in the second column, datasets can also be downloaded via dataset name. Column N is the total number of instances in the dataset. Column J is the number of classes in the target variable, here, 0 denotes regression problems. C is the count of the categorical columns and P is the count of numerical columns in the dataset. In columns seven to nine, mean cardinality of the categorical columns ($\frac{\sum_{c=1}^{C} K_c}{C}$), total cardinality of the categorical columns ($\sum_{c=1}^{C} K_c$) and missing percentage of the dataset (NA (%)) are reported respectively. For all datasets, if the number of instances is greater than 50000, the dataset is sampled down to 50000 instances. Number of the instances of the original datasets range between 736 and 163065. Number of classes for multiclass datasets range between four and five. 33.3% of the datasets only consist of categorical

features and do not contain any numerical features. The number of categorical features in the datasets range between one and 71. Total count of cardinality of the datasets ranges between five and 67478. Total cardinality is calculated by taking the sum of the cardinality of categorical variables in the dataset. Mean cardinality of the datasets ranges between 3.56 and 13495.60. Mean cardinality is calculated by taking the mean of the cardinality of categorical columns in the dataset. Missing values are present in 46% of the datasets. Missing percentage of datasets range between 0% and 18.84%.

Table 5.1. Dataset descriptions.

ID	Name	N	J	C	P	$\frac{\sum_{c=1}^{C} K_c}{C}$	$\sum_{c=1}^{C} K_c$	NA (%)
43898	adult	48790	2	9	6	11.22	101	0.88
43900	amazon_employee_access	32769	2	10	0	1562.80	15628	0.00
43901	click_prediction_small	39926	2	5	4	13495.60	67478	0.00
43920	kdd_internet_usage	10108	2	71	0	8.30	589	0.50
43902	kick	72983	2	13	20	167.46	2177	0.42
43922	mushroom	8124	2	22	0	5.36	118	0.00
41442	open_payments	73354	2	6	0	1432.33	8594	18.84
1461	bank-marketing	45211	2	10	7	4.60	46	0.00
43938	nursery	12960	5	9	0	3.56	32	0.00
188	eucalyptus	736	5	6	14	13.67	82	3.04
41212	hpc-job-scheduling	4331	4	3	5	8.33	25	0.00
43939	california_housing	20640	0	1	9	5.00	5	0.10
41444	medical_charges	163065	0	5	2	1127.00	5635	0.00
43926	ames_housing	2930	0	46	35	6.91	318	0.00
43927	avocado_sales	18249	0	2	12	28.00	56	0.00

5.3. Machine Learning Algorithms

Hyperparameter tuning is not performed for the machine learning models used, considering that the aim is to compare the performance of the encoding strategies. For each encoding strategy, its performance is ranked for a single machine learning algorithm at a time. Hyperparameter tuning can later be performed to improve the predictive performance of a model with the best categorical encoding rank. 10-fold cross-validation with five repetitions is performed to split the dataset. For classification setting stratified cross-validation is used. Mean of 50 cross-validation scores yield the final mean score.

Logistic Regression model is trained with stats R package for binary classification setting. Logistic Regression model is trained with nnet R package [34] for multiclass classification setting. Linear Regression model is trained with R stats package. Random Forests were trained with the ranger R package [35]. XGBoost models are trained with xgboost R package. Objective functions used are binary:logistic (logistic regression for binary classification), multi:softprob (softmax objective for multiclass classification) and reg:squarederror (squared loss for regression) [36].

For evaluation of binary classification setting, Area Under the ROC Curve (AUC) is calculated. For multiclass classification setting, multiclass-AUC is calculated with pROC R package [37]. For regression setting, mean absolute percentage error (MAPE) is calculated.

6. RESULTS

In this chapter, how existing encoding strategies and our methodology perform in several real-world datasets is introduced. First, encoding strategies are benchmarked and their performance estimates are displayed in boxplots, second, rankings of the encodings are estimated, third, datasets are clustered based on cardinality, fourth, effect of increasing tree complexity on categorical split encoding is investigated, fifth, the robustness of the supervised encoding strategies are provided as missing values are randomly generated at various percentages. Lastly, the number of columns generated by encoding strategies are provided.

6.1. Comparison of Encoding Strategies

Performance estimates that show minimum, mean, maximum and outlier values for 10 fold cross-validation and five repetitions (average of 50 results for each dataset), of the encoding strategies are provided in Figures 6.1 (regression), 6.2 (multiclass classification) and 6.3 (binary classification). For Figure 6.1 (regression) y-axis is reversed for ease of interpretation, as a lower mean score of MAPE indicates better performance of the encoding strategy. In Appendix A, mean performance estimates of the datasets are provided in Tables A.1, A.2, A.3. OHE cannot be performed for some datasets that contain high-cardinality categorical features. Target encoding outperforms categorical split encoding on majority of the datasets. However, the variance of categorical split encoding across cross-validation folds is lower compared to target encoding and the other encoding strategies. The best performance is achieved by target encoding followed by categorical split encoding. One-hot encoding, when applicable, was an efficient strategy. Frequency and leaf encoding performances comparable. Integer encoding often displayed the worst performance out of all encoding strategies. Categorical split encoding often performed well with high-cardinality datasets, where OHE cannot be performed due to memory issues. For datasets that contain an ordinal categorical feature, such as kick dataset, it is better to use target encoding or inte-

ger encoding. Categorical split encoding often performs better with generalized linear models compared to other encoding strategies. Target encoding performs better with random forest and xgboost algorithms.

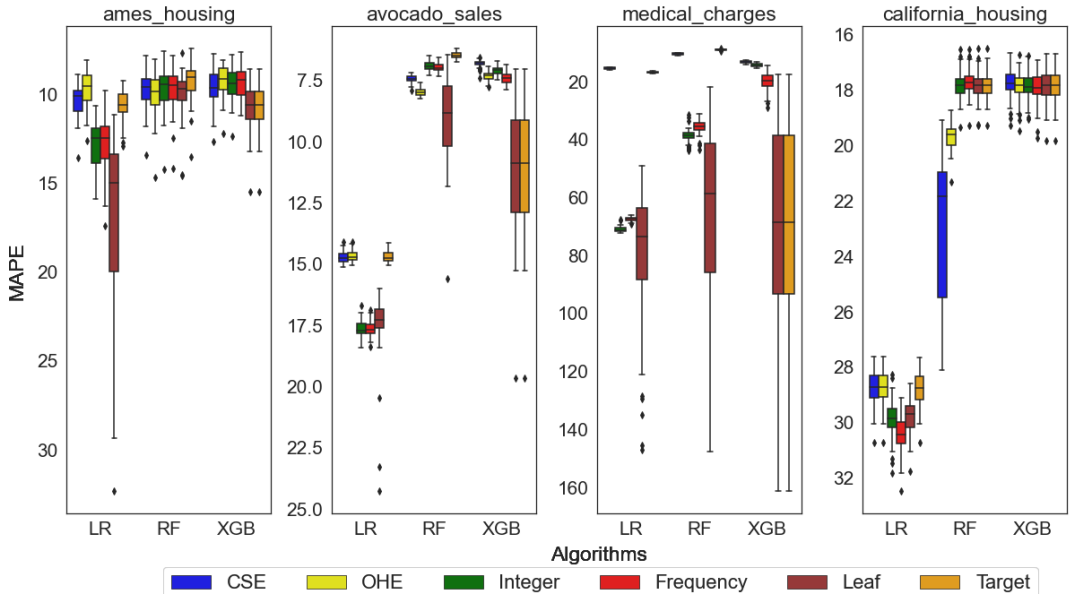

Figure 6.1. Performance estimates with 10 fold cv and five repetitions for regression datasets.

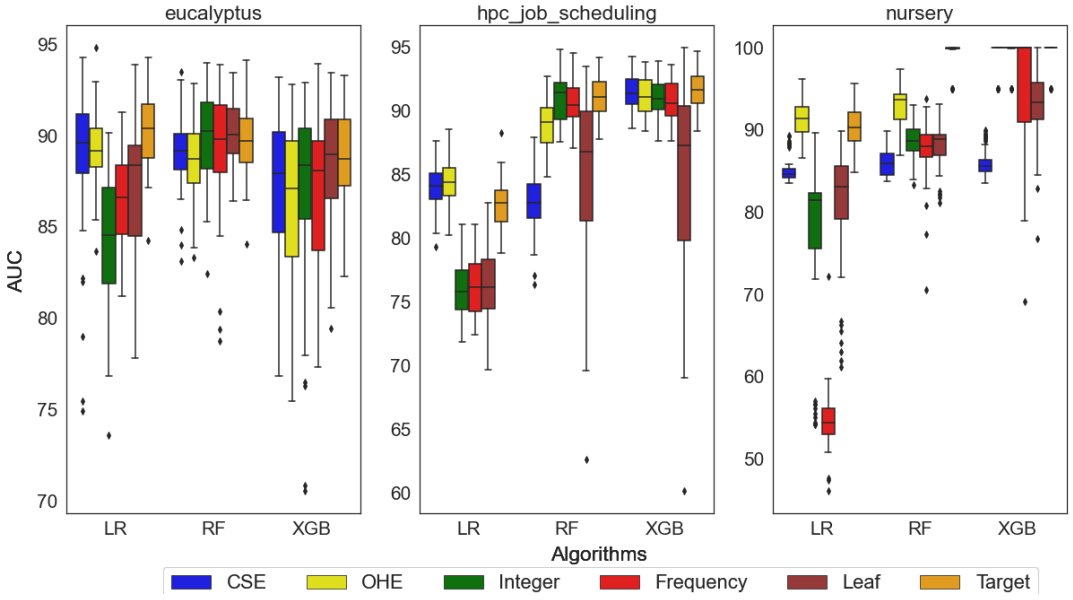

Figure 6.2. Performance estimates with 10 fold cv and five repetitions for multiclass classification datasets.

Performances for binary classification datasets are provided in Figure 6.3. Variance of categorical split encoding is lower compared to other encoding strategies. One-hot encoding is not performed for amazon_employee_access, click_prediction_small, kick and open_payments datasets as it is computationally demanding.

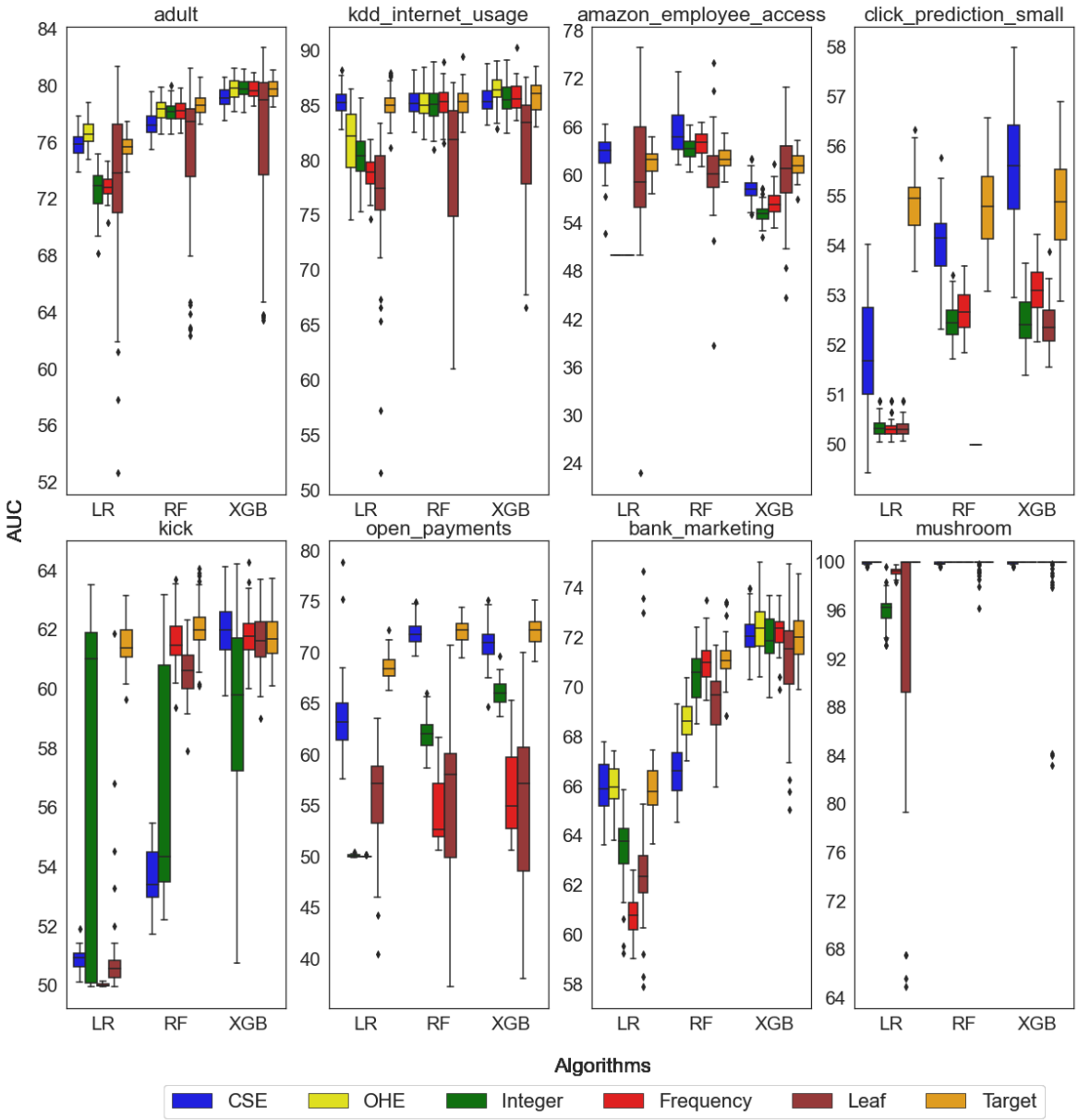

Figure 6.3. Performance estimates with 10 fold cv and five repetitions for binary classification datasets.

6.2. Rank Based Comparison

To rank the encoding strategies, for each dataset, ranks over each cross-validation fold is computed. Then, initially, Friedman's test is performed to check whether there is a significant difference between the encoding strategies at 0.05 significance level [38]. Based on Friedman's test, there was a statistically significant difference between the encoding strategies at 0.05 significance level, meaning at least one of the encoding strategies differed from the rest. Then, Nemenyi post-hoc test is performed to check which encoding strategies differ. Critical difference diagram is plotted using Nemenyi test, by computing mean ranks across all cross-validation folds and datasets [39]. Critical difference plots are displayed in Figures 6.4 and 6.5 when ranks are computed based on OHE applicability. Since, for some datasets, one-hot encoding cannot be performed, two critical difference diagrams are provided. Figure 6.4 for datasets that one-hot encoding was performed on and Figure 6.5 for all datasets excluding one-hot encoding results. In Figure 6.4, that displays the critical difference diagram for OHE applicable datasets, critical difference value is 2.01 at 0.05 significance level. Horizontal lines indicate statistically non-significant methods. Target encoding ranks the best followed by one-hot encoding, however, it can observed that the ranks of the encoding methods are not statistically different except for target encoding and leaf encoding.

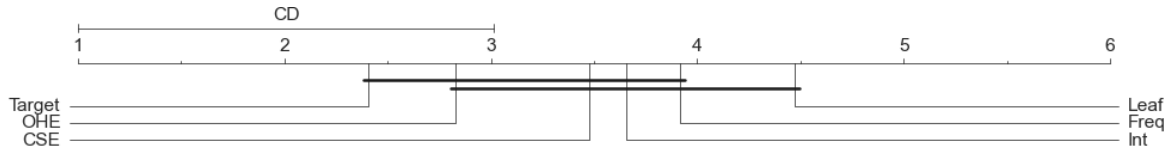

Figure 6.4. Critical difference diagram for all encoding strategies (OHE applicable datasets).

In Figure 6.5, critical difference diagram for all datasets excluding one-hot encoding is displayed. Critical difference value is 1.57 at 0.05 significance level, similar to the results in Figure 6.4, target encoding ranks the best followed by categorical split encoding. Integer encoding ranks third, followed by frequency encoding and leaf encoding respectively. However, encoding methods are not significantly different except

for target encoding and leaf encoding. Hence, the ranks are not statistically significant except for target encoding and leaf encoding.

Figure 6.5. Critical difference diagram excluding OHE results.

Ranks are also computed based on the type of algorithm used, namely, whether generalized linear models, random forest or xgboost is performed. Critical difference diagrams on algorithm level are provided in Figures 6.6, 6.7 and 6.8. For each of the figures, critical difference value is 1.52 at 0.05 significance level. In Figure 6.6 target encoding ranks the best followed by CSE. Target encoding and CSE are not significantly different, but the other encoding stratigies are significanly different than target encoding and CSE. In Figures 6.7 and 6.8, critical difference diagrams for random forest and xgboost algorithms are provided respectively. Target encoding ranks as the best strategy in all critical difference diagrams. Categorical split encoding ranks fourth in Figure 6.7, in random forest critical diagram.

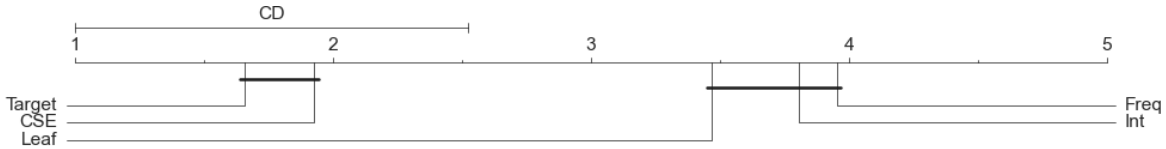

Figure 6.6. Critical difference diagram of generalized linear models.

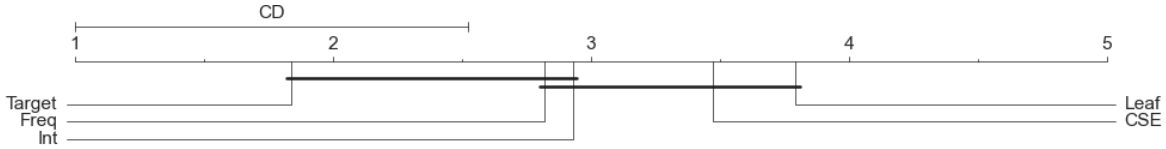

Figure 6.7. Critical difference diagram of random forest.

Categorical split encoding ranks third in Figure 6.8, in xgboost critical diagram. Leaf encoding ranks the worst out of all encoding strategies. However, encoding strategies are not significantly different from one another.

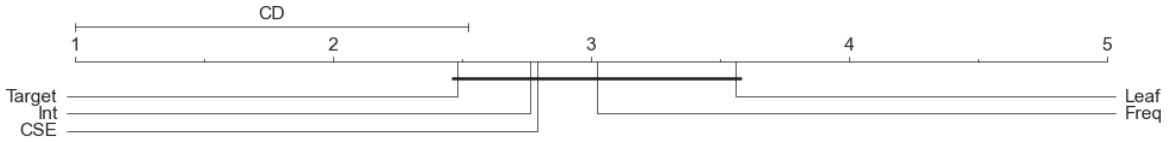

Figure 6.8. Critical difference diagram of xgboost.

6.3. Dataset Clustering

Datasets are clustered based on their cardinalities. First, a dataset is generated that consists of five columns, that describes the cardinality of the datasets. Cardinality information of datasets for hierarchical clustering is given in Table 6.1. First column is the name of the dataset. Columns two to six are the number of categorical columns in the dataset (C), maximum cardinality of the dataset ($\max(K_c)$), mean cardinality of the dataset ($\frac{\sum_{c=1}^{C} K_c}{C}$), total cardinality of the dataset ($\sum_{c=1}^{C} K_c$) and ratio of the max cardinality of the dataset and the number of instances in the dataset ($\max(K_c)/N$) respectively.

Table 6.1. Cardinality information dataset.

Name	C	$\max(K_c)$	$\frac{\sum_{c=1}^{C} K_c}{C}$	$\sum_{c=1}^{C} K_c$	$\max(K_c)/N$
adult	9	41	11.22	101	0.08
amazon_employee_access	10	7518	1562.80	15628	22.94
click_prediction_small	5	22381	13495.60	67478	56.06
kdd_internet_usage	71	129	8.30	589	1.28
kick	13	1063	167.46	2177	1.46
mushroom	22	12	5.36	118	0.15
open_payments	6	4362	1432.33	8594	5.95
bank-marketing	10	12	4.60	46	0.03
nursery	9	5	3.56	32	0.04
eucalyptus	6	27	13.67	82	3.67
hpc-job-scheduling	3	14	8.33	25	0.32
california_housing	1	5	5.00	5	0.02
medical_charges	5	3201	1127.00	5635	1.96
ames_housing	46	28	6.91	318	0.96
avocado_sales	2	54	28.00	56	0.30

For ease of interpretation, the number of clusters is set to four. Elbow method plot, to determine the optimal number of clusters, is provided in Figure 6.9. Optimal number of clusters is determined to be four and shown on the plot with a vertical dashed line. Total within sum of square is used for estimating the optimal number of clusters.

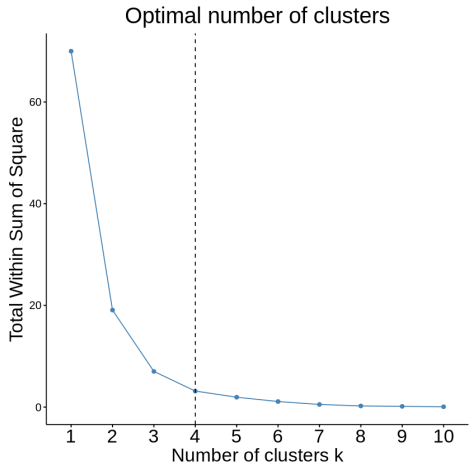

Figure 6.9. Optimal number of clusters with elbow method.

Since, the dataset in Table 6.1 only consists of 15 instances, it can be considered a small dataset, hierarchical clustering is performed as the method of choice. The dataset is scaled and distance matrix is computed using euclidean distance. Hierarchical clustering model is fit using ward linkage method. The dendrogram visualization of hierarchical clustering is provided in Figure 6.10 and how the datasets nest in clusters is visually observed.

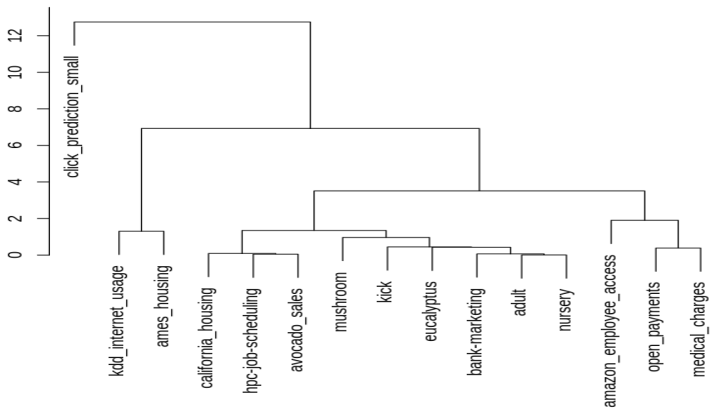

Figure 6.10. Hierarchical clustering dendrogram with ward linkage.

The clusters that datasets fall into is given in Table 6.2, in the first column of the table. OHE applicable datasets fall into cluster one and four except kick dataset in cluster one. Cluster three consists of only *click_prediction* dataset which is a high-cardinality dataset. Cluster one consists of the low-cardinality datasets, cluster four consists of low-medium-cardinality datasets and cluster two consists of medium-cardinality datasets. To compare the performance of the encoding strategies among clusters, rank of the encoding strategies is also provided in Table 6.2 for generalized linear models. Datasets that OHE cannot be applied for are marked with - in OHE column. The best ranks for each encoding strategy are marked in bold. Ranks are computed over 10 fold cross-validation and five repetitions. It is observed that categorical split encoding performs better with low-medium-cardinality to high-cardinality datasets, namely, with those datasets in clusters two, three and four. Target encoding and OHE performs better with low-cardinality datasets, namely, with datasets in cluster one. CSE, OHE and target encoding mostly rank better compared to frequency encoding, integer encoding and leaf encoding when used with generalized linear models.

Table 6.2. Rank based comparison of encoding strategies for generalized linear models.

cluster	Name	CSE	Freq	Int	Leaf	OHE	Target
1	adult	2.54	5.12	5.14	3.78	**1.32**	3.10
1	avocado_sales	2.46	5.34	5.18	4.42	**1.18**	2.08
1	bank-marketing	1.98	5.90	4.26	4.66	**1.88**	2.30
1	california_housing	1.52	5.98	4.48	4.52	**1.20**	2.76
1	eucalyptus	2.54	4.34	5.28	3.96	3.00	**1.84**
1	hpc-job-scheduling	1.80	4.64	5.22	5.12	**1.24**	2.90
1	kick	2.94	4.76	2.32	3.36	-	**1.52**
1	mushroom	2.56	4.58	5.58	3.76	**1.88**	**1.88**
1	nursery	3.20	5.90	4.78	3.98	**1.26**	1.84
2	amazon_employee_access	**1.54**	3.98	3.98	2.30	-	2.24
2	medical_charges	**1.00**	3.32	4.42	4.26	-	2.00
2	open_payments	2.04	4.28	4.26	3.14	-	**1.04**
3	click_prediction_small	2.24	3.86	3.64	3.62	-	**1.00**
4	ames_housing	2.14	4.52	4.70	5.70	**1.26**	2.62
4	kdd_internet_usage	**1.54**	4.92	4.12	5.22	3.46	1.74

The rank and clusters of the encoding strategies for random forest is provided in Table 6.3. For medium and high-cardinality datasets which belong to clusters two and three, categorical split encoding performs better compared to datasets in other

clusters. Frequency and target encoding rank better compared to other strategies, for datasets in cluster one.

Table 6.3. Rank based comparison of encoding strategies for random forest.

cluster	Name	CSE	Freq	Int	Leaf	OHE	Target
1	adult	5.44	3.24	3.34	4.56	2.88	**1.48**
1	avocado_sales	4.2	2.8	2.38	5.26	5.3	**1**
1	bank-marketing	5.78	1.86	2.84	3.98	4.8	**1.7**
1	california_housing	5.98	**1.52**	2.5	2.78	5.02	2.68
1	eucalyptus	4.12	3.48	2.8	**2.7**	4.8	3.08
1	hpc-job-scheduling	5.64	2.7	2.12	4.38	4.24	**1.92**
1	kick	4.74	1.96	3.94	3.3	-	**1.06**
1	mushroom	4.52	**2.9**	**2.9**	3.7	**2.9**	**2.9**
1	nursery	5.52	4.16	3.94	4.08	2.3	**1**
2	amazon_employee_access	**1.64**	2.3	3	4.18	-	3.86
2	medical_charges	2	3.34	4.06	4.6	-	**1**
2	open_payments	1.68	4.56	3.12	4.28	-	**1.36**
3	click_prediction_small	1.92	3.22	3.74	5	-	**1.08**
4	ames_housing	3.5	3.5	3.4	4.2	5.12	**1.08**
4	kdd_internet_usage	3	3.14	3.6	5.4	**2.8**	3.02

The rank of the encoding strategies for xgboost is provided in Table 6.4. Categorical split encoding performs best with medium and high-cardinality datasets in clusters two and three. For OHE applicable datasets target encoding and OHE rank better, such as datasets in cluster one.

Table 6.4. Rank based comparison of encoding strategies for xgboost.

cluster	Name	CSE	Freq	Int	Leaf	OHE	Target
1	adult	5.22	3	2.88	4.26	**2.72**	2.84
1	avocado_sales	**1.06**	3.8	2.06	4.76	3.52	4.76
1	bank-marketing	3.46	2.84	3.88	4.44	**2.72**	3.54
1	california_housing	**2.92**	3.98	3.56	3.02	3.48	3.02
1	eucalyptus	3.6	3.72	3.76	2.98	4.14	**2.74**
1	hpc-job-scheduling	2.76	4.36	3.42	5.12	2.94	**2.32**
1	kick	**2.28**	2.7	3.96	3.08	-	2.94
1	mushroom	4.28	**2.88**	**2.88**	3.68	**2.88**	**2.88**
1	nursery	5.76	3.26	2.4	4.62	2.06	**1.88**
2	amazon_employee_access	2.92	3.74	4.58	2.1	-	**1.66**
2	medical_charges	**1.02**	3.06	2	3.96	-	3.96
2	open_payments	1.86	4.5	3.04	4.4	-	**1.18**
3	click_prediction_small	**1.28**	3.06	4.28	4.54	-	1.82
4	ames_housing	3.54	2.46	2.72	4.6	**2.06**	4.6
4	kdd_internet_usage	3.72	3.26	3.46	5.32	**2.02**	3.16

6.4. Effect of Increasing Tree Complexity on Categorical Split Encoding

In this section, the effect of increasing tree complexity is explored. Maxdepth parameter of the tree is chosen as the parameter to control tree complexity. As discussed, increasing maxdepth parameter of a decision tree yields a more complex model, that captures more details about the training data. However, above a certain depth (this depth varies for different datasets), overfitting occurs and this leads to a model that seems to work efficiently on training set but does not perform well on the test set. For categorical split encoding, maxdepth is set to four. For each cluster, discussed in section 6.3, the effect of increasing maxdepth is provided with one dataset in that cluster.

The effect of increasing maxdepth (4, 5, 6, 7) for adult, amazon_employee_access, click_prediction_small and kdd_internet_usage datasets is observed in Figure 6.11. The results in tabular form are provided in Appendix B in Table B.1. For adult dataset as maxdepth increases between four and six, auc score also increases for random forest and xgboost algorithms. At depth seven auc score drops. Auc score increases constantly with increasing maxdepth, with generalized linear models. For click_prediction_small dataset, increasing maxdepth does not effect the performance of categorical split encoding for any algorithm. For kdd_internet_usage dataset, auc score does not change with increasing maxdepth for random forest and xgboost algorithms, decreases with generalized linear models. For amazon_employee_access dataset, performance does not change with xgboost algorithm, with increasing maxdepth. For linear and random forest algorithms, auc score fluctuates and does not linearly increase.

It is observed that, performance does not improve constantly, with increasing maxdepth. The change in performance depends on the dataset. A functioning relationship between the cluster a dataset resides in and the performance as the maxdepth increases is not observed when performing categorical split encoding. While the performance in some datasets, such as adult, can improve between maxdepths four and six, performance in some datasets, such as kdd_internet_usage, can show a decrease.

One important thing to note is that increasing maxdepth also increases computation time for categorical split encoding. Hence, trying various maxdepth options when performing categorical split encoding can be considered but computation time should also be taken into consideration.

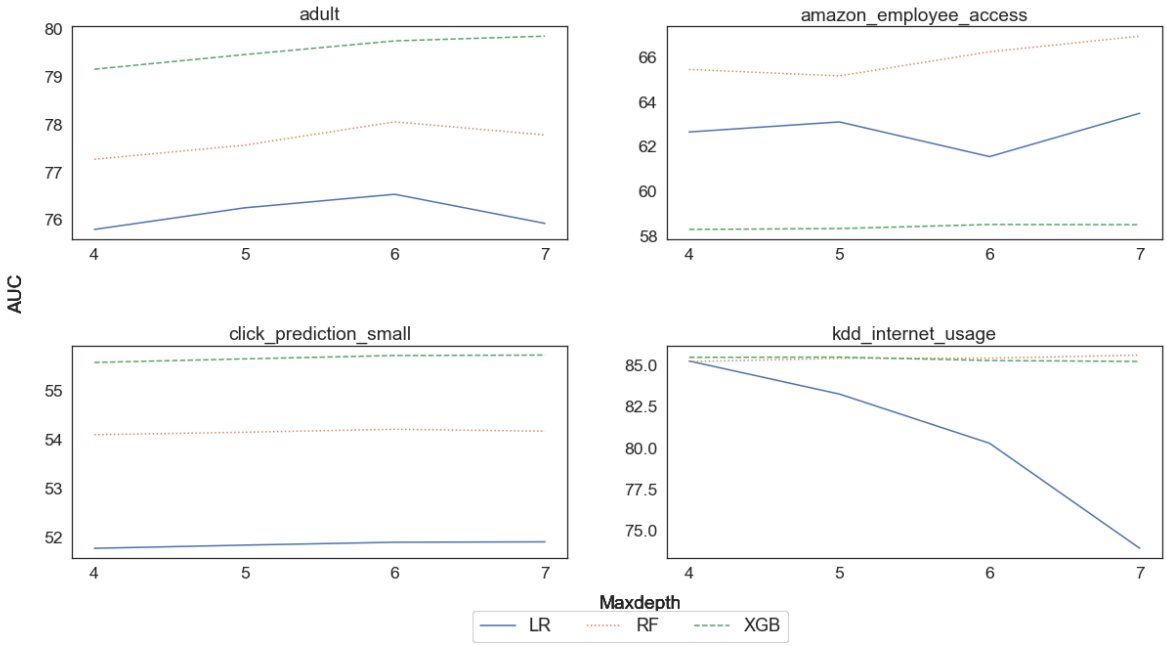

Figure 6.11. Effect of increasing maxdepth on CSE.

6.5. Generation of Missing Values Completely at Random

Missing values are generated at completely at random for four missing rates for each dataset. Four missing rates are set as 10%, 20%, 30%, 40%. If the categorical column had already more missing percentage than the missing percentage set, no missing values are generated for that column. If the categorical column has less missing percentage than the set one then, missing rate is set as the difference of the set missing percentage and missing percentage of the categorical column. Change in the performances of categorical split encoding, leaf encoding and target encoding are observed with increasing missing rate. Plots for classification and regression problems are observed for generalized linear models, random forest and xgboost algorithms separately. The results for the generalized linear models are provided in Figure 6.12 for classification datasets. Results for random forest and xgboost algorithms are provided

in Appendix C. Categorical split encoding and target encoding results are robust while leaf encoding fluctuates with increasing missing rate and the performance of it decreases.

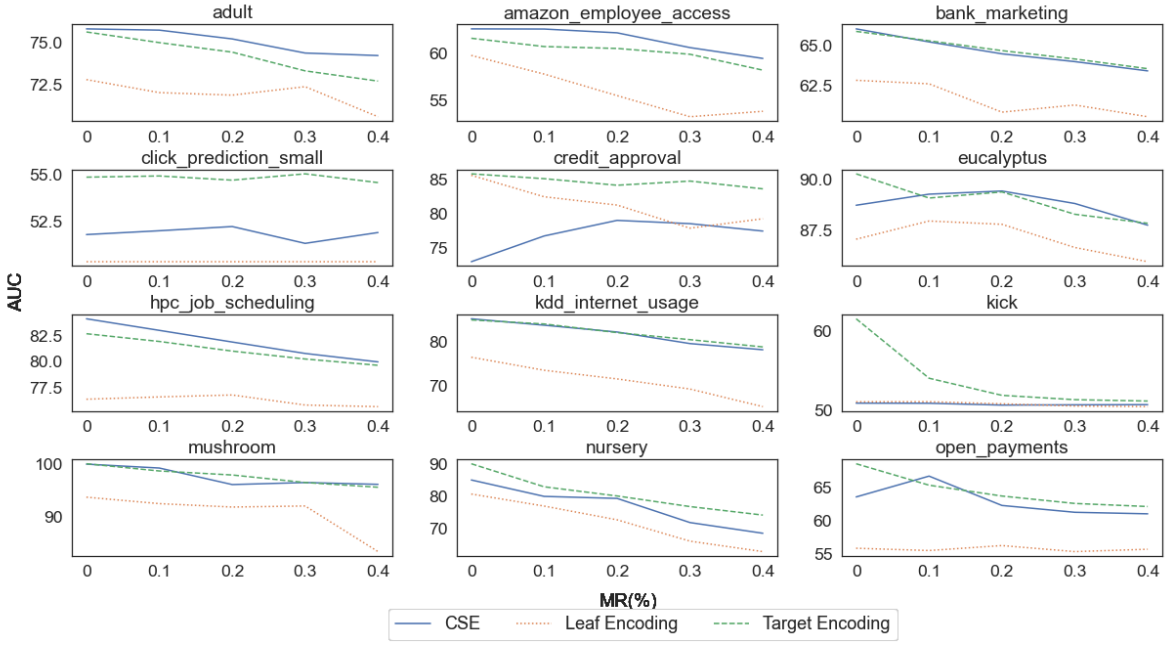

Figure 6.12. Generating missing values at various missing rates classification results for logistic regression.

The results for the generalized linear models are provided in Figure 6.13 for regression datasets. For regression problems, y-axis is reversed for ease of interpretation. Hence, a decrease in Figure 6.13 indicates increase in MAPE score, which is interpreted as decrease in performance. Categorical split encoding and target encoding performances are robust in the presence of increasing percentage of missing values. Performance of leaf encoding decreases with increasing missing percentage in most cases. Another thing to note is that the initial performance of leaf encoding is worse compared to categorical split encoding and target encoding when missing values are not created completely at random. In ames_housing a slight increase in performance is observed between missing rates 0% and 20% with leaf encoding. In medical_charges dataset a slight increase in performance is observed between missing rates 20% and 30%. In some cases, such as for ames_housing dataset when leaf encoding is performed increasing missing percentage between 0% and 20%, an increase in performance with

increasing missing percentage is observed. This can be attributed to generation of missing values completely at random.

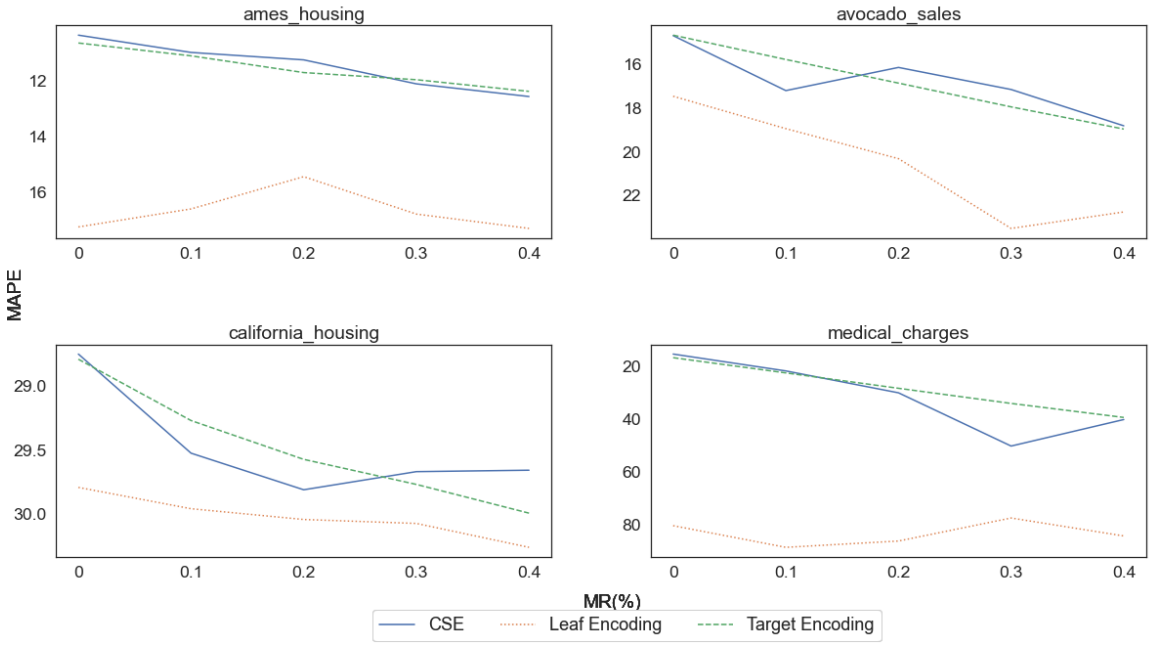

Figure 6.13. Generating missing values at various missing rates regression results for linear regression.

6.6. Number of Columns Generated by The Methods

In Table 6.5 the number of columns generated by categorical split encoding, one-hot encoding, leaf encoding and target encoding is given. Frequency and integer encoding are not included as they do not yield any additional columns. Target encoding only creates additional columns for multiclass classification datasets. Categorical split encoding yields less columns compared to one-hot encoding and can be used for all datasets without causing memory problems. Due to averaging of cross validation folds, number of columns generated by categorical split encoding are not integer values. Values written in italic in OHE column, are the datasets OHE cannot be performed for. Since one-hot encoding generates as many columns as the total cardinality of the dataset, the values in OHE column are equal to the total cardinality of the datasets. As observed, datasets with total cardinality over 2177 are not one-hot encoding applicable,

as it takes too long to run. Leaf encoding sometimes generates a single column because a split is not made on the categorical variable.

Table 6.5. Number of columns generated.

	CSE	**OHE**	**Leaf**	**Target**
adult	26.28	101	7	9
amazon_employee_access	201.40	*15628*	1	10
bank_marketing	37.60	46	8	10
click_prediction_small	88.32	*67478*	5	5
eucalyptus	64.34	82	15	20
hpc_job_scheduling	37.42	25	6	6
kdd_internet_usage	142.80	589	1	71
kick	81.78	*2177*	21	13
mushroom	28.50	118	1	22
nursery	25.00	32	1	32
open_payments	92.60	*8594*	1	6
ames_housing	86.10	318	35	46
avocado_sales	27.96	56	12	2
california_housing	11.90	5	9	1
medical_charges	209.12	*5635*	2	5

7. CONCLUSION

This thesis proposes a novel method for encoding categorical variables, which uses split information of a decision tree as the encoding of that categorical column. Categorical split encoding reduces the dimensionality of the categorical columns, providing computational ease and generates less features compared to OHE. It makes use of the innate tree schema of the dataset. Surrogate information of a categorical variable serves as an important key for imputing missing values. While clustering similar values together based on the path they take through the decision tree algorithm, missing values are encoded as well.

Alongside the novel method presented, predictive performance of a variety of encoding strategies on different settings have been benchmarked. It is concluded that smoothed target encoding mostly outperforms other encoding strategies, however has high variance over cross-validation folds, whereas categorical split encoding also is as efficient and has low variance. Using OHE and target encoding for low-cardinality datasets is shown to be better. For medium and high-cardinality variables using categorical split encoding and target encoding is advised. For ordinal categorical variables use of integer encoding yields good results. However, other than ordinal categorical variables use of integer encoding yields poor results. Categorical split encoding works well with generalized linear models compared to other encoding strategies. It can perform with high-cardinality variables, whereas OHE cannot. Another advantage is encoding missing values innately, in real-world datasets often contain attributes with a high percentage of missing values and categorical split encoding has proven to be robust in presence of high percentage of missing values.

For future work, it can be beneficial to use an ensemble of all or any subset of categorical encoding methods discussed in this thesis. An improvement to the method can be finding an optimal hyperparameter setting for categorical split encoding by experimenting with various hyperparameters and benchmarking the results. Especially

finding an optimum way that will maximize the evaluation score of categorical split encoding, to select maxdepth for each dataset is of concern. Using randomized ensemble of multiple trees instead of using a single decision tree is a promising avenue of research.

REFERENCES

1. Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, Vol. 12, No. 1, pp. 2825–2830, 2011.

2. Perlich, C. and F. Provost, "Distribution-Based Aggregation for Relational Learning with Identifier Attributes", *Machine Learning*, Vol. 62, No. 1, pp. 65–105, 2006.

3. Arat, M. M., "Learning From High-Cardinality Categorical Features in Deep Neural Networks", *Journal of Advanced Research in Natural and Applied Sciences*, Vol. 8, No. 2, pp. 222–236, 2022.

4. Moeyersoms, J. and D. Martens, "Including High-Cardinality Attributes in Predictive Models: A Case Study in Churn Prediction in the Energy Sector", *Decision Support Systems*, Vol. 72, No. 1, pp. 72–81, 2015.

5. Cerda, P., G. Varoquaux and B. Kégl, "Similarity Encoding for Learning with Dirty Categorical Variables", *Machine Learning*, Vol. 107, No. 8, pp. 1477–1494, 2018.

6. Cerda, P. and G. Varoquaux, "Encoding High-Cardinality String Categorical Variables", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 34, No. 3, pp. 1164–1176, 2022.

7. Boriah, S., V. Chandola and V. Kumar, "Similarity Measures for Categorical Data: A Comparative Evaluation", *Proceedings of the 2008 SIAM International Conference on Data Mining*, pp. 243–254, Georgia, USA, 2008.

8. Pargent, F., F. Pfisterer, J. Thomas and B. Bischl, "Regularized Target Encoding Outperforms Traditional Methods in Supervised Machine Learning with High Cardinality Features", *Computational Statistics*, Vol. 1, No. 1, pp. 1–22, 2022.

9. Hancock, J. T. and T. M. Khoshgoftaar, "Survey on Categorical Data for Neural Networks", *Journal of Big Data*, Vol. 7, No. 1, pp. 1–41, 2020.

10. Seca, D. and J. Mendes-Moreira, "Benchmark of Encoders of Nominal Features for Regression", *World Conference on Information Systems and Technologies*, pp. 146–155, Terceira Island, Portugal, 2021.

11. Wright, M. N. and I. R. König, "Splitting on Categorical Predictors in Random Forests", *PeerJ*, Vol. 7, No. 1, p. e6339, 2019.

12. Alamuri, M., B. R. Surampudi and A. Negi, "A Survey of Distance/Similarity Measures for Categorical Data", *International Joint Conference on Neural Networks (IJCNN)*, pp. 1907–1914, Beijing, China, 2014.

13. Carrizosa, E., M. G. Restrepo and D. R. Morales, "On Clustering Categories of Categorical Predictors in Generalized Linear Models", *Expert Systems with Applications*, Vol. 182, No. 1, p. 115245, 2021.

14. Breiman, L., J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees*, Routledge, 2017.

15. Babinec, T., "CHAID Response Modeling and Segmentation", *Quirck's Marketing Research Review*, Vol. 1, No. 1, pp. 12–15, 1990.

16. Lucena, B., "Exploiting Categorical Structure Using Tree-Based Methods", *International Conference on Artificial Intelligence and Statistics*, pp. 2949–2958, Sicily, Italy, 2020.

17. Cohen, P., S. G. West and L. S. Aiken, *Applied Multiple Regression/Correlation*

Analysis for the Behavioral Sciences, Psychology Press, 2014.

18. Nukoolkit, C., H. Chen and D. Brown, "A Data Transformation Technique for Car Injury Prediction", *Computer Science Department, University of Alabama*, Vol. 1, No. 1, pp. 5–7, 2001.

19. Johnson, S. C., "Hierarchical Clustering Schemes", *Psychometrika*, Vol. 32, No. 3, pp. 241–254, 1967.

20. Pargent, F., B. Bischl and J. Thomas, *A Benchmark Experiment on How to Encode Categorical Features in Predictive Modeling*, Master's Thesis, Ludwig-Maximilians-Universität München, 2019.

21. Weinberger, K., A. Dasgupta, J. Langford, A. Smola and J. Attenberg, "Feature Hashing for Large Scale Multitask Learning", *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1113–1120, Quebec, Canada, 2009.

22. Kuhn, M. and K. Johnson, *Feature Engineering and Selection: A Practical Approach for Predictive Models*, CRC Press, 2019.

23. Choong, A. C. H. and N. K. Lee, "Evaluation of Convolutionary Neural Networks Modeling of DNA Sequences Using Ordinal Versus One-Hot Encoding Method", *International Conference on Computer and Drone Applications (IConDA)*, pp. 60–65, Sarawak, Malaysia, 2017.

24. Von Eye, A. and C. C. Clogg, *Categorical Variables in Developmental Research: Methods of Analysis*, Elsevier, 1996.

25. Quinlan, J. R., "Induction of Decision Trees", *Machine Learning*, Vol. 1, No. 1, pp. 81–106, 1986.

26. Micci-Barreca, D., "A Preprocessing Scheme for High-Cardinality Categorical At-

tributes in Classification and Prediction Problems", *ACM SIGKDD Explorations Newsletter*, Vol. 3, No. 1, pp. 27–32, 2001.

27. Prokhorenkova, L., G. Gusev, A. Vorobev, A. V. Dorogush and A. Gulin, "CatBoost: Unbiased Boosting with Categorical Features", *Advances in Neural Information Processing Systems*, Vol. 31, No. 1, pp. 1–23, 2018.

28. Loh, W.-Y., "Classification and Regression Trees", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 1, No. 1, pp. 14–23, 2011.

29. Song, Y.-Y. and L. Ying, "Decision Tree Methods: Applications for Classification and Prediction", *Shanghai Archives of Psychiatry*, Vol. 27, No. 2, p. 130, 2015.

30. Team, R. C., "R: A Language and Environment for Statistical Computing", *R Foundation for Statistical Computing*, Vol. 1, No. 1, p. 1, 2013.

31. Gazioğlu, M. and M. Baydoğan, "Categorical Split Encoding", 2021, `https://github.com/minegazioglu/CatSplitEnc`, accessed on July 20, 2022.

32. Kuhn, M., "Building Predictive Models in R Using the Caret Package", *Journal of Statistical Software*, Vol. 28, No. 1, pp. 1–26, 2008.

33. Vanschoren, J., J. N. van Rijn, B. Bischl and L. Torgo, "OpenML: Networked Science in Machine Learning", *SIGKDD Explorations*, Vol. 15, No. 2, pp. 49–60, 2013.

34. Venables, W. N. and B. D. Ripley, *Modern Applied Statistics with S*, Springer, New York, 2002.

35. Wright, M. N. and A. Ziegler, "Ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R", *Journal of Statistical Software*, Vol. 77, No. 1, pp. 1–17, 2017.

36. Chen, T. and C. Guestrin, "Xgboost: A Scalable Tree Boosting System", *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, New York, USA, 2016.

37. Robin, X., N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.-C. Sanchez and M. Müller, "PROC: An Open-Source Package for R and S+ to Analyze and Compare ROC Curves", *BMC Bioinformatics*, Vol. 12, No. 1, p. 77, 2011.

38. Friedman, M., "A Comparison of Alternative Tests of Significance for the Problem of M Rankings", *The Annals of Mathematical Statistics*, Vol. 11, No. 1, pp. 86–92, 1940.

39. Nemenyi, P. B., *Distribution-Free Multiple Comparisons.*, Princeton University, 1963.

APPENDIX A: MEAN CROSS-VALIDATION RESULTS

In this section, mean results of the 10 fold cross-validation and five repetitions (50 scores for each dataset) is supplied for classification and regression datasets. For some datasets, OHE is not performed as they were computationally demanding.

Table A.1. Mean multiclass AUC scores for multiclass classification datasets.

		LR	**RF**	**XGB**
eucalyptus	**Categorical Split Encoding**	88.74	89.06	87.46
	Frequency Encoding	86.68	89.34	86.70
	Integer Encoding	84.19	89.93	86.62
	Leaf Encoding	87.09	90.14	88.38
	Target Encoding	90.27	89.75	88.92
	OHE	89.24	88.69	86.32
hpc_job_scheduling	**Categorical Split Encoding**	84.12	82.81	91.47
	Frequency Encoding	76.19	90.59	90.74
	Integer Encoding	75.94	90.98	91.09
	Leaf Encoding	76.38	85.03	84.71
	Target Encoding	82.68	91.13	91.59
	OHE	84.45	88.99	91.28
nursery	**Categorical Split Encoding**	85.16	86.11	86.08
	Frequency Encoding	54.82	87.46	95.11
	Integer Encoding	75.92	88.66	98.99
	Leaf Encoding	80.75	87.97	93.11
	Target Encoding	90.24	98.96	99.00
	OHE	91.14	93.06	98.99

For regression datasets, a lower MAPE score indicates a better performance in terms of encoding strategy and algorithm used.

Table A.2. Mean MAPE scores for regression datasets.

		LR	**RF**	**XGB**
ames_housing	**Categorical Split Encoding**	10.36	9.72	9.67
	Frequency Encoding	12.82	9.76	9.33
	Integer Encoding	12.81	9.74	9.40
	Leaf Encoding	17.23	9.98	10.69
	Target Encoding	10.64	9.30	10.69
	OHE	9.68	9.99	9.23
avocado_sales	**Categorical Split Encoding**	14.70	7.43	6.79
	Frequency Encoding	17.64	6.96	7.38
	Integer Encoding	17.63	6.91	7.06
	Leaf Encoding	17.47	9.13	11.10
	Target Encoding	14.68	6.46	11.10
	OHE	14.67	7.96	7.31
california_housing	**Categorical Split Encoding**	28.74	22.89	17.78
	Frequency Encoding	30.44	17.74	17.95
	Integer Encoding	29.84	17.81	17.85
	Leaf Encoding	29.79	17.85	17.84
	Target Encoding	28.78	17.83	17.84
	OHE	28.73	19.70	17.87
medical_charges	**Categorical Split Encoding**	15.32	10.35	13.07
	Frequency Encoding	67.36	35.47	19.90
	Integer Encoding	70.82	38.74	14.24
	Leaf Encoding	80.40	65.10	70.80
	Target Encoding	16.70	8.90	70.80

Mean area under the curve scores are given for eight binary classification datasets. OHE is not performed for kick, click_prediction_small and open_payment datasets.

Table A.3. Mean AUC scores for binary classification datasets.

		LR	RF	XGB
adult	**Categorical Split Encoding**	75.80	77.27	79.16
	Frequency Encoding	72.88	78.21	79.76
	Integer Encoding	72.51	78.16	79.76
	Leaf Encoding	72.79	74.91	76.10
	Target Encoding	75.62	78.63	79.77
	OHE	76.63	78.27	79.78
amazon_employee_access	**Categorical Split Encoding**	62.66	65.47	58.30
	Frequency Encoding	50.00	63.88	56.47
	Integer Encoding	50.00	63.24	55.21
	Leaf Encoding	59.83	60.43	60.65
	Target Encoding	61.66	62.01	61.21
bank_marketing	**Categorical Split Encoding**	66.02	66.62	72.07
	Frequency Encoding	60.80	71.04	72.27
	Integer Encoding	63.47	70.44	71.98
	Leaf Encoding	62.85	69.41	70.99
	Target Encoding	65.86	71.13	72.08
	OHE	66.02	68.69	72.36
click_prediction_small	**Categorical Split Encoding**	51.78	54.10	55.57
	Frequency Encoding	50.33	52.67	53.11
	Integer Encoding	50.33	52.47	52.48
	Leaf Encoding	50.33	50.00	52.40
	Target Encoding	54.85	54.79	54.87
kdd_internet_usage	**Categorical Split Encoding**	85.29	85.26	85.51
	Frequency Encoding	78.89	85.22	85.74
	Integer Encoding	80.51	84.96	85.67
	Leaf Encoding	76.49	79.11	80.84
	Target Encoding	85.03	85.25	85.80
	OHE	81.70	85.26	86.41
kick	**Categorical Split Encoding**	50.88	53.65	61.99
	Frequency Encoding	50.03	61.61	61.84
	Integer Encoding	57.10	56.88	59.30
	Leaf Encoding	51.05	60.57	61.65
	Target Encoding	61.50	62.10	61.80
mushroom	**Categorical Split Encoding**	99.93	99.90	99.91
	Frequency Encoding	99.22	100.00	100.00
	Integer Encoding	95.94	100.00	100.00
	Leaf Encoding	93.69	99.75	98.85
	Target Encoding	100.00	100.00	100.00
	OHE	100.00	100.00	100.00
open_payments	**Categorical Split Encoding**	63.57	71.83	70.87
	Frequency Encoding	50.01	54.31	56.22
	Integer Encoding	50.07	61.95	66.03
	Leaf Encoding	55.83	55.39	55.37
	Target Encoding	68.58	72.02	71.99

APPENDIX B: RESULTS OF INCREASING MAXDEPTH

Effect of increasing maxdepth for one dataset in each cluster is given in Table B.1. To represent cluster four the result for *kdd_internet_usage* dataset, to represent cluster three the result for *click_prediction_small* dataset is given. To represent cluster two, the result of *amazon_employee_access* dataset is provided. To represent cluster one *adult* dataset is used.

Table B.1. Change in CSE performance with increasing maxdepth.

		LR	RF	XGB
adult	Categorical Split Encoding_4	75.80	77.27	79.16
	Categorical Split Encoding_5	76.26	77.57	79.47
	Categorical Split Encoding_6	76.54	78.06	79.75
	Categorical Split Encoding_7	75.93	77.78	79.85
	Frequency Encoding	72.88	78.21	79.76
	Integer Encoding	72.51	78.16	79.76
	Leaf Encoding	72.79	74.91	76.10
	OHE	76.63	78.27	79.78
	Target Encoding	75.62	78.63	79.77
amazon_employee_access	Categorical Split Encoding_4	62.66	65.47	58.30
	Categorical Split Encoding_5	63.12	65.18	58.33
	Categorical Split Encoding_6	61.56	66.26	58.52
	Categorical Split Encoding_7	63.50	66.96	58.51
	Frequency Encoding	50.00	63.88	56.47
	Integer Encoding	50.00	63.24	55.21
	Leaf Encoding	59.83	60.43	60.65
	Target Encoding	61.66	62.01	61.21
click_prediction_small	Categorical Split Encoding_4	51.78	54.10	55.57
	Categorical Split Encoding_5	51.85	54.15	55.65
	Categorical Split Encoding_6	51.90	54.21	55.72
	Categorical Split Encoding_7	51.91	54.17	55.73
	Frequency Encoding	50.33	52.67	53.11
	Integer Encoding	50.33	52.47	52.48
	Leaf Encoding	50.33	50.00	52.40
	Target Encoding	54.85	54.79	54.87
kdd_internet_usage	Categorical Split Encoding_4	85.29	85.26	85.51
	Categorical Split Encoding_5	83.28	85.45	85.53
	Categorical Split Encoding_6	80.29	85.46	85.32
	Categorical Split Encoding_7	73.90	85.65	85.26
	Frequency Encoding	78.89	85.22	85.74
	Integer Encoding	80.51	84.96	85.67
	Leaf Encoding	76.49	79.11	80.84
	Target Encoding	85.03	85.25	85.80

APPENDIX C: GENERATION OF MISSING VALUES

In this chapter, various percentages of missing values are generated completely random for all datasets and the performance of CSE, Target Encoding and Leaf Encoding are observed.

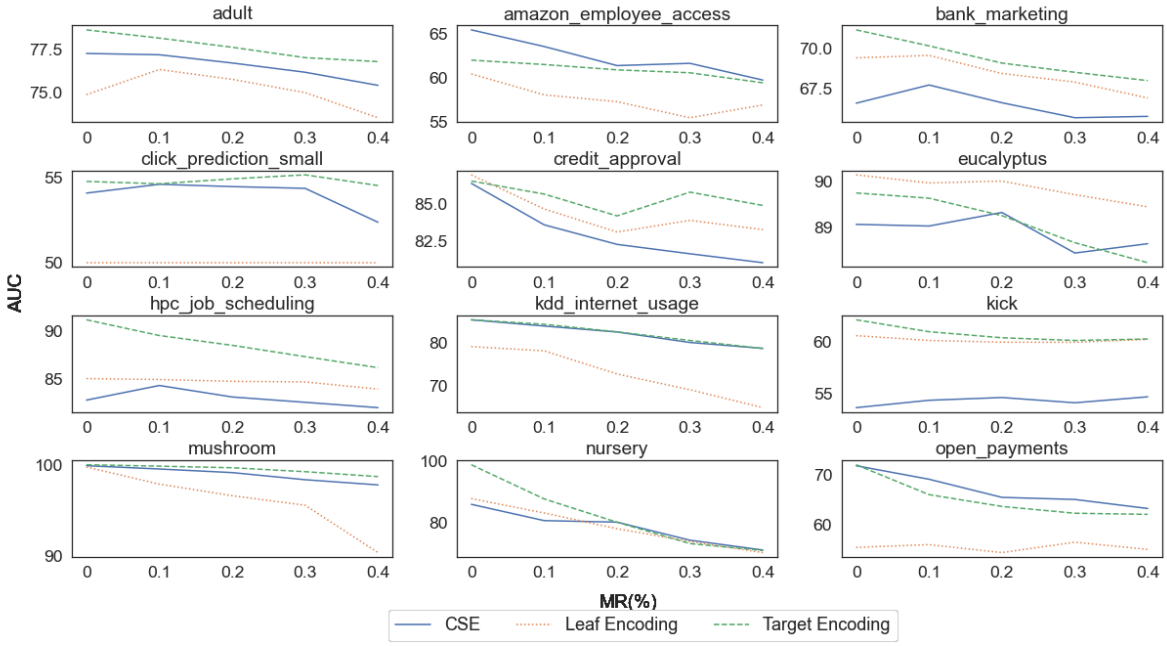

Figure C.1. Generating missing values at various missing rates classification results for random forest.

Performances of target encoding and CSE are robust in the presence of increasing percentage of missing values, while leaf encoding's performance decreases in general.

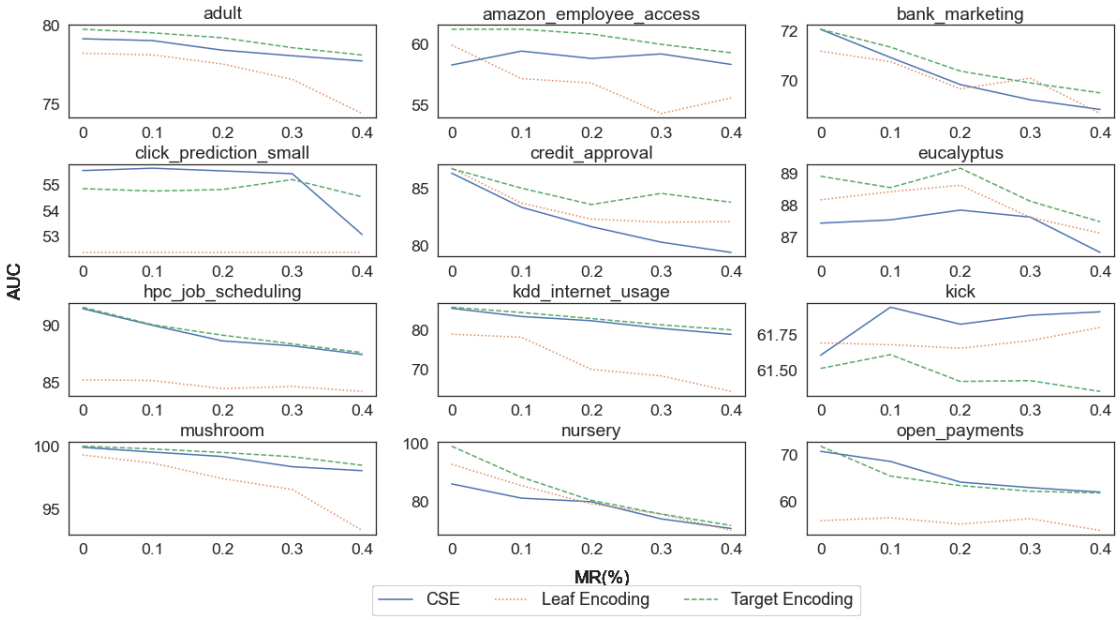

Figure C.2. Generating missing values at various missing rates classification results for xgboost.

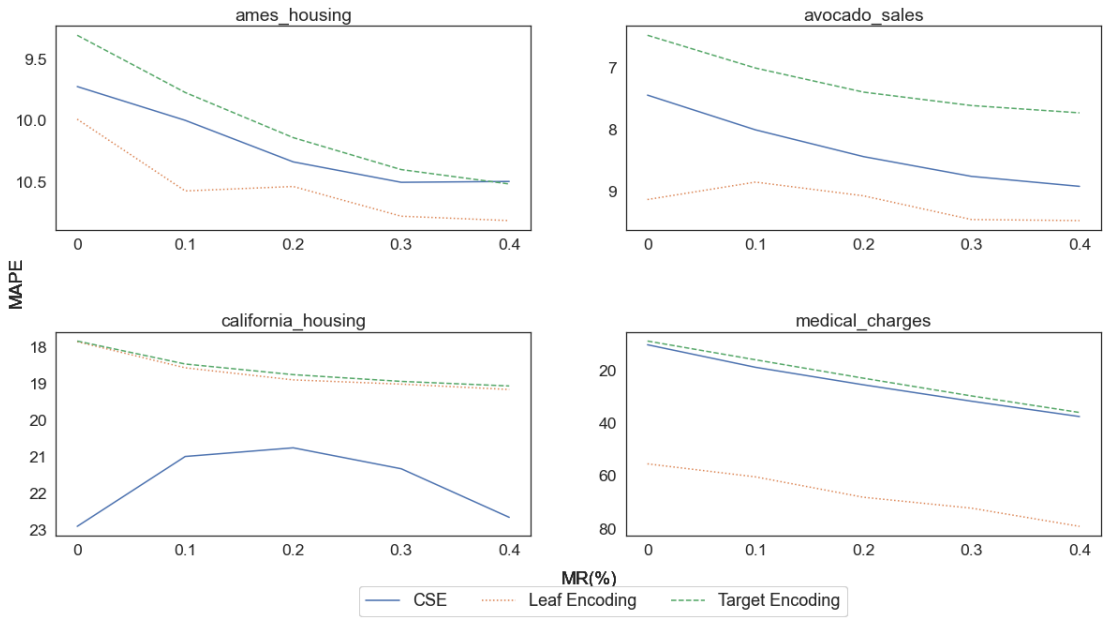

Figure C.3. Generating missing values at various missing rates regression results for random forest.

For regression datasets y-axis is reversed for ease of interpretation. Hence, a decrease in Figure C, means an increase in MAPE, indicating decrease in performance.

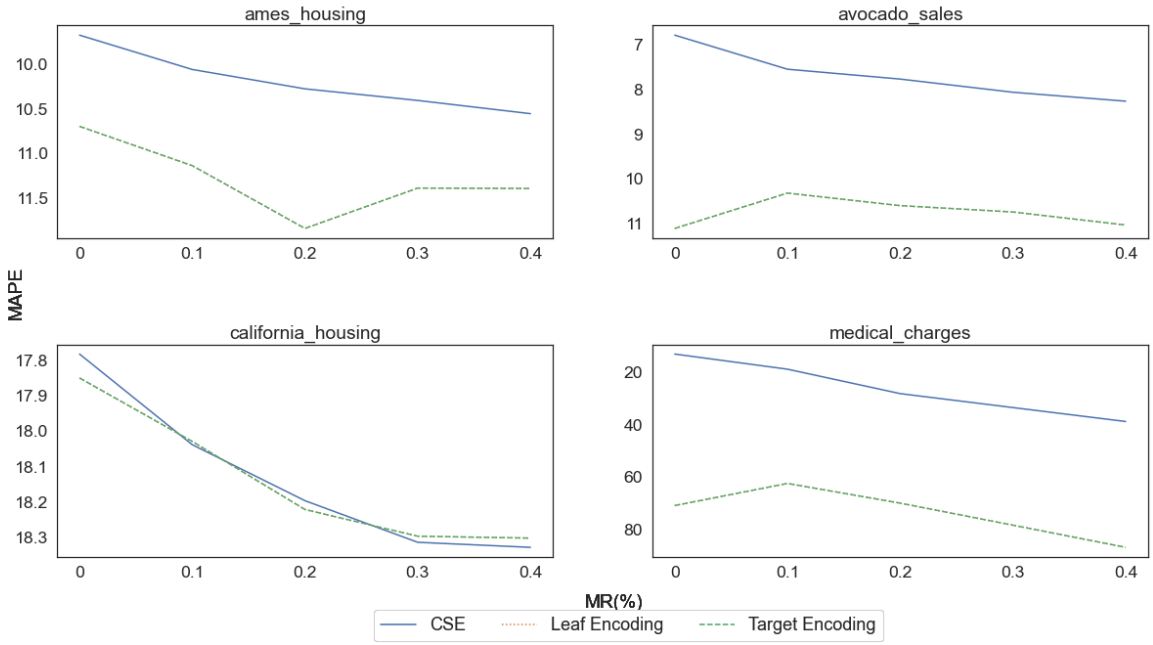

Figure C.4. Generating missing values at various missing rates regression results for xgboost.